

CPEN 455: Deep Learning

Lecture 7: Graph Neural Networks

Renjie Liao

University of British Columbia

Winter, Term 1, 2026

Outline

- Applications of Graphs
- Background
- Challenges of Deep Learning on Graphs
- GNNs
 - Overview
 - Message Passing
 - Architectures and Attention
 - Readout
- Implementation
- Relationship w. Transformers

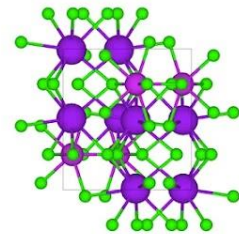
Outline

- **Applications of Graphs**
- Background
- Challenges of Deep Learning on Graphs
- GNNs
 - Overview
 - Message Passing
 - Architectures and Attention
 - Readout
- Implementation
- Relationship w. Transformers

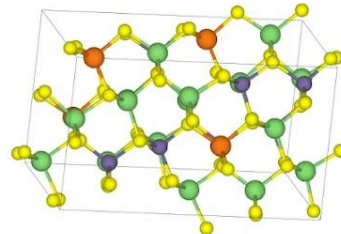
Motivating Applications of Graphs

Materials discovery: predict the stability of a crystal [14].

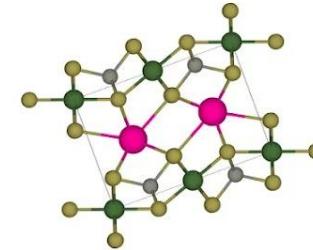
Nodes: atoms. Edges: nearby atomic interactions. Target: a graph-level property.



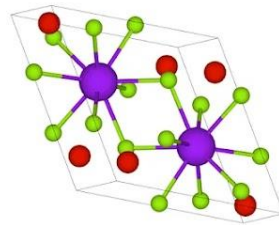
K_2BiCl_5



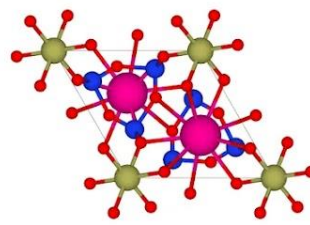
$\text{Li}_4\text{MgGe}_2\text{S}_7$



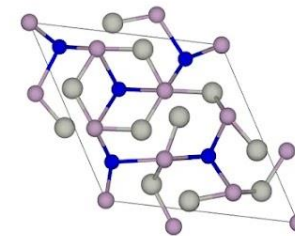
Mo_5GeB_2



KV_3Se_3



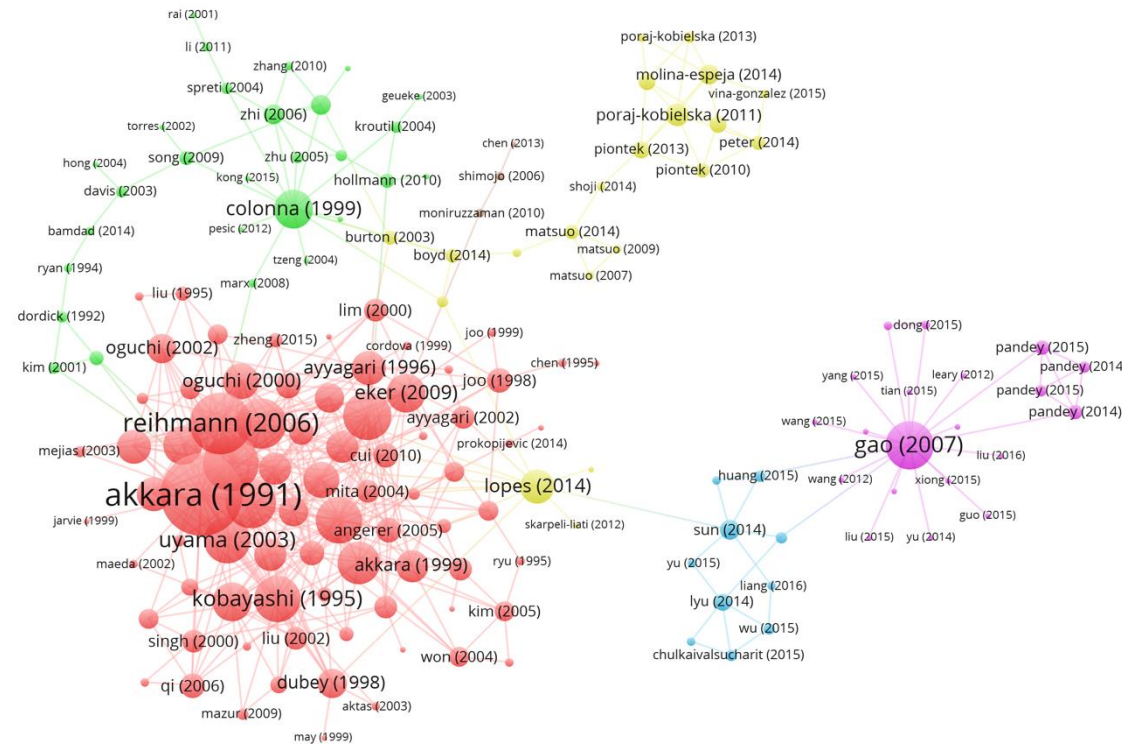
$\text{Rb}_2\text{HfSi}_3\text{O}_9$



$\text{Tm}_5\text{Pd}_9\text{P}_7$

Motivating Applications of Graphs

- Citation Networks: Predict Each Paper's Topic

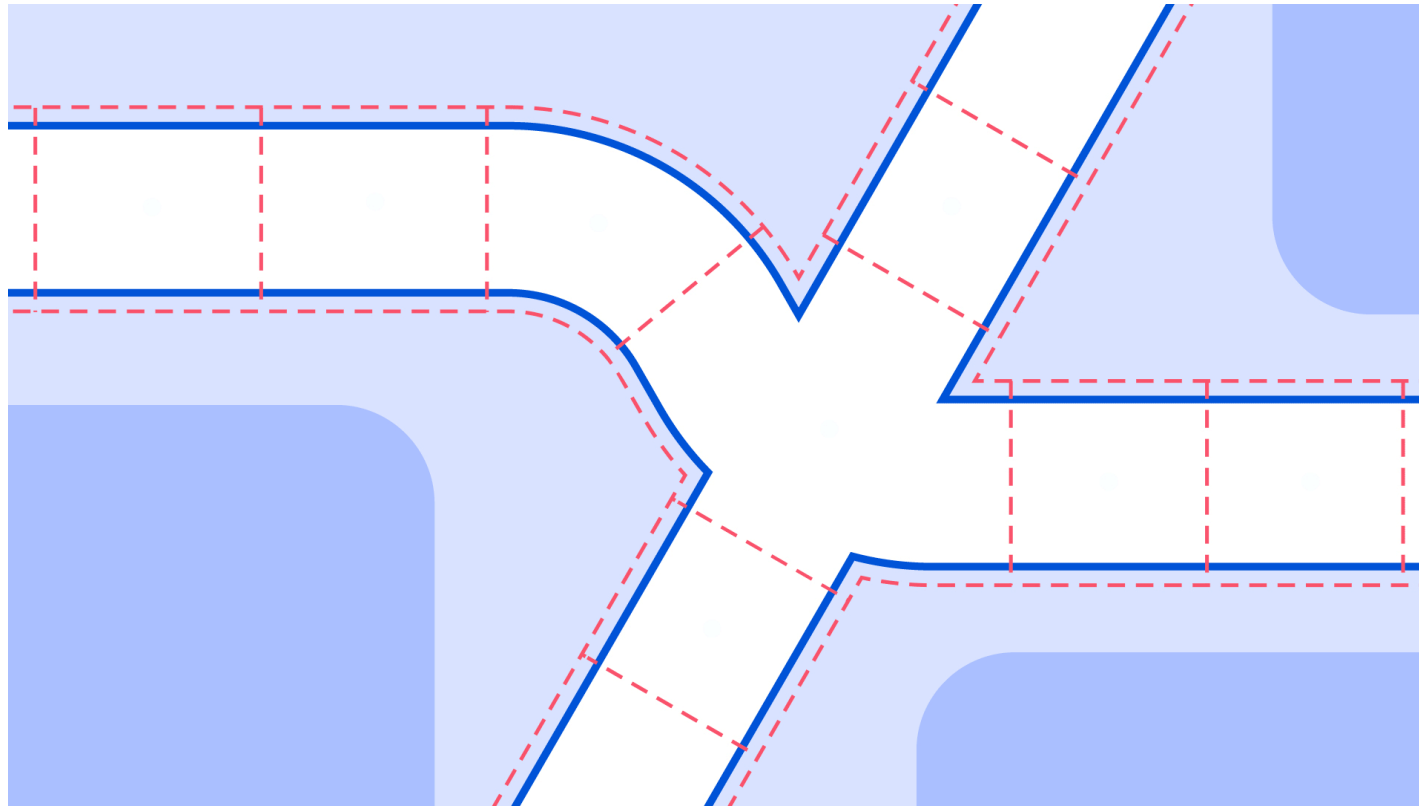


Nodes are papers; citation links supply context for node classification.

Motivating Applications of Graphs

Traffic prediction: use connected road segments to predict travel time [16].

Nodes: road segments. A road subgraph supplies the travel-time target.



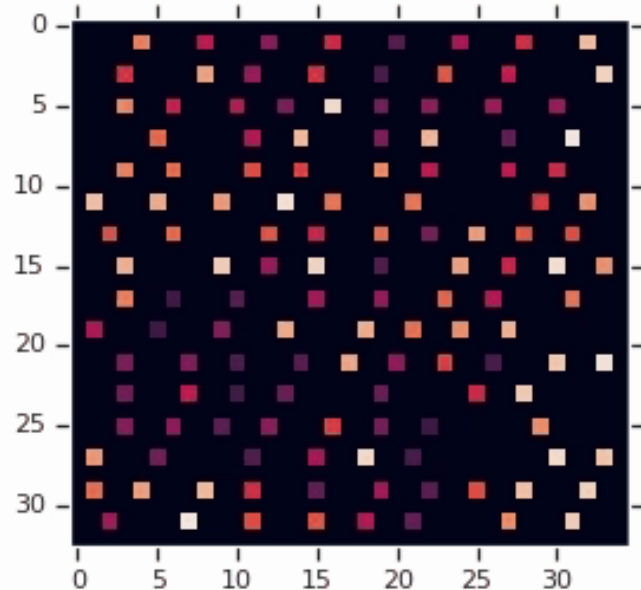
Motivating Applications of Graphs

Chip design: place interconnected circuit components on a chip [21].

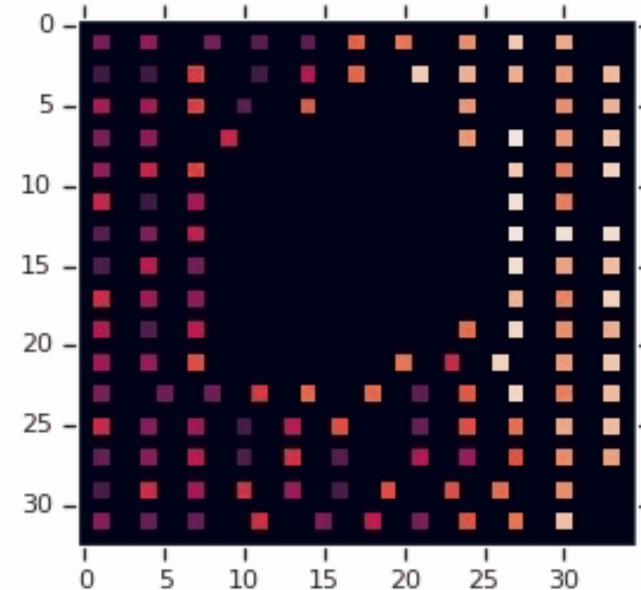
Nodes: circuit components. Edges: electrical connections.

AlphaChip combines a GNN with reinforcement learning to choose placements.

Without pre-training



After pre-training



Outline

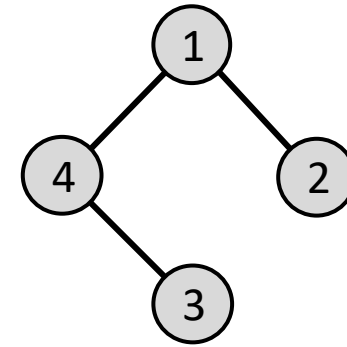
- Applications of Graphs
- **Background**
- Challenges of Deep Learning on Graphs
- GNNs
 - Overview
 - Message Passing
 - Architectures and Attention
 - Readout
- Implementation
- Relationship w. Transformers

Deep Learning on Graphs

Graph Representations

- Connectivity

1. Edge List: $G = (V, E)$



$$V = \{1, 2, 3, 4\}, E = \{(1, 2), (1, 4), (4, 3)\}$$

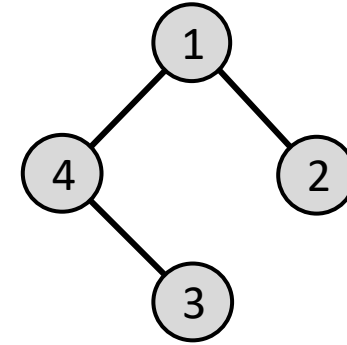
Deep Learning on Graphs

Graph Representations

- Connectivity

1. Edge List: $G = (V, E)$

2. Adjacency Matrix: A (sometimes we have weights)



$V = \{1,2,3,4\}, E = \{(1,2), (1,4), (4,3)\}$

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Deep Learning on Graphs

Graph Representations

- Connectivity

1. Edge List: $G = (V, E)$

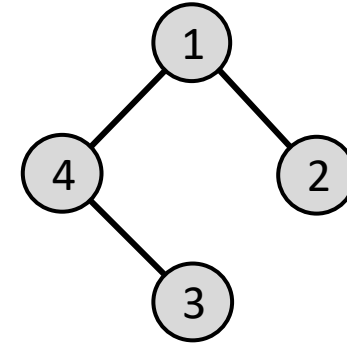
2. Adjacency Matrix: A (sometimes we have weights)

- Feature

1. Node Feature: X

2. Edge Feature

3. Graph Feature



$V = \{1,2,3,4\}, E = \{(1,2), (1,4), (4,3)\}$

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Deep Learning on Graphs

Graph Representations

- Connectivity

1. Edge List: $G = (V, E)$

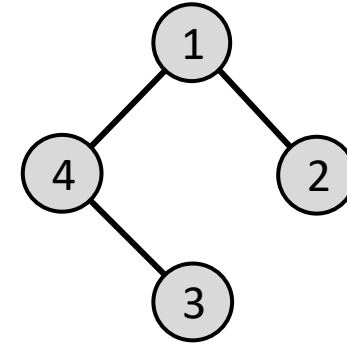
2. Adjacency Matrix: A (sometimes we have weights)

- Feature

1. Node Feature: X

2. Edge Feature

3. Graph Feature



$V = \{1,2,3,4\}, E = \{(1,2), (1,4), (4,3)\}$

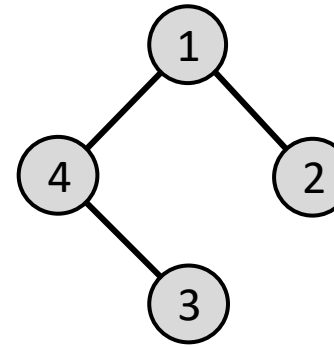
	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Node-attributed graph: (A, X)

Deep Learning on Graphs

Permutation

$$\begin{array}{lll} V = [1,2,3,4] & \Rightarrow & V' = [2,1,3,4] \\ E = [(1,2), (1,4), (4,3)] & \Rightarrow & E' = [(2,1), (2,4), (4,3)] \end{array}$$



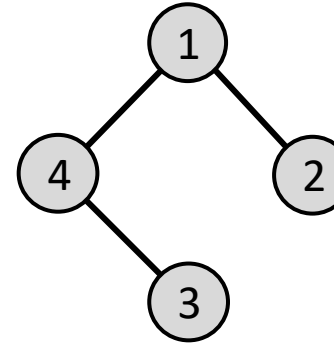
	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

Deep Learning on Graphs

Permutation

$$\begin{array}{ll} V = [1,2,3,4] & \Rightarrow V' = [2,1,3,4] \\ E = [(1,2), (1,4), (4,3)] & \Rightarrow E' = [(2,1), (2,4), (4,3)] \end{array}$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Original Adj Matrix

Deep Learning on Graphs

Permutation

$$V = [1, 2, 3, 4]$$

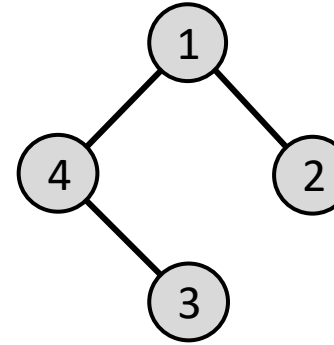
$$E = [(1, 2), (1, 4), (4, 3)]$$

$\Rightarrow$

$$V' = [2, 1, 3, 4]$$

$\Rightarrow$

$$E' = [(2, 1), (2, 4), (4, 3)]$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1, 2, 3, 4], E = [(1, 2), (1, 4), (4, 3)]$$

Permute Rows

	1	2	3	4
1	0	1	0	0
2	1	0	0	0
3	0	0	1	0
4	0	0	0	1

Permutation Matrix

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Original Adj Matrix

Permute Columns

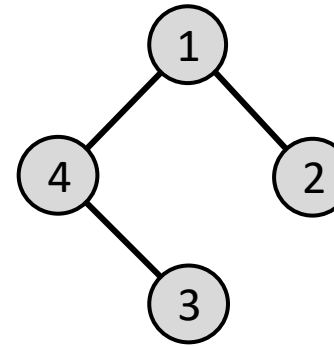
	1	2	3	4
1	0	1	0	0
2	1	0	0	0
3	0	0	1	0
4	0	0	0	1

Transposed
Permutation Matrix

Deep Learning on Graphs

Permutation

$$\begin{aligned} V &= [1, 2, 3, 4] & \Rightarrow & & V' &= [2, 1, 3, 4] \\ E &= [(1, 2), (1, 4), (4, 3)] & \Rightarrow & & E' &= [(2, 1), (2, 4), (4, 3)] \end{aligned}$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1, 2, 3, 4], E = [(1, 2), (1, 4), (4, 3)]$$

Permute Rows

	1	2	3	4
1	0	1	0	0
2	1	0	0	0
3	0	0	1	0
4	0	0	0	1

Permutation Matrix

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Original Adj Matrix

Permute Columns

	1	2	3	4
1	0	1	0	0
2	1	0	0	0
3	0	0	1	0
4	0	0	0	1

Transposed
Permutation Matrix

=

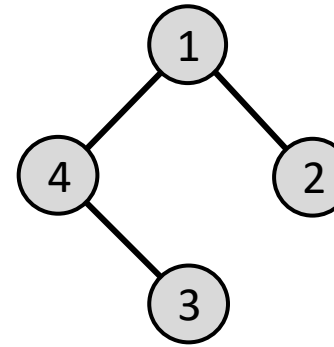
	1	2	3	4
1	0	1	0	0
2	1	0	0	1
3	0	0	0	1
4	0	1	1	0

Permuted Adj Matrix

Deep Learning on Graphs

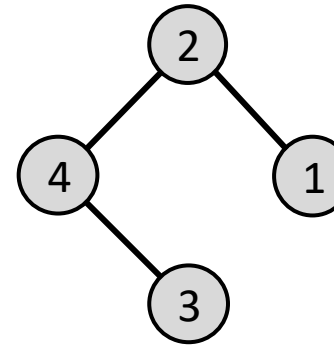
Permutation

$$\begin{array}{lll} V = [1,2,3,4] & \Rightarrow & V' = [2,1,3,4] \\ E = [(1,2), (1,4), (4,3)] & \Rightarrow & E' = [(2,1), (2,4), (4,3)] \end{array}$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$



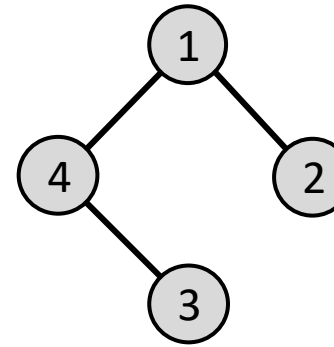
	1	2	3	4
1	0	1	0	0
2	1	0	0	1
3	0	0	0	1
4	0	1	1	0

$$V' = [2,1,3,4], E' = [(2,1), (2,4), (4,3)]$$

Deep Learning on Graphs

Permutation

$$\begin{array}{lll} V = [1,2,3,4] & \Rightarrow & V' = [2,1,3,4] \\ E = [(1,2), (1,4), (4,3)] & \Rightarrow & E' = [(2,1), (2,4), (4,3)] \end{array}$$



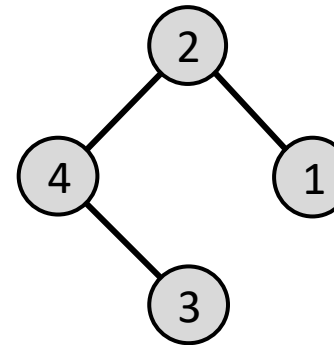
	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

Graph Isomorphism:

A graph isomorphism is a bijection of nodes that preserves adjacency.

$$PA_1P^T = A_2$$



	1	2	3	4
1	0	1	0	0
2	1	0	0	1
3	0	0	0	1
4	0	1	1	0

$$V' = [2,1,3,4], E' = [(2,1), (2,4), (4,3)]$$

Deep Learning on Graphs

Permutation Invariance & Equivariance

Graph Data (A, X) , Model $f(A, X)$

Invariance:

$$f(PAP^{\top}, PX) = f(A, X)$$

Equivariance:

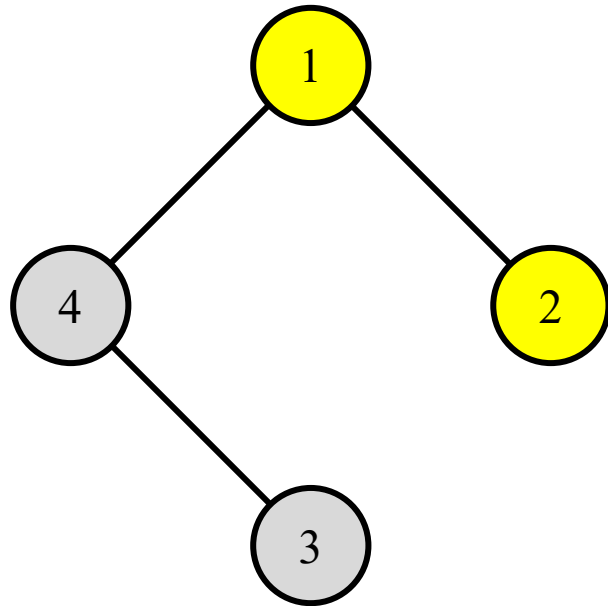
$$f(PAP^{\top}, PX) = Pf(A, X)$$

Permutation Invariance and Equivariance

Shared node functions and order-invariant aggregation preserve node equivariance.

$$F(PAP^{\top}, P\mathbf{H}) = PF(A, \mathbf{H})$$

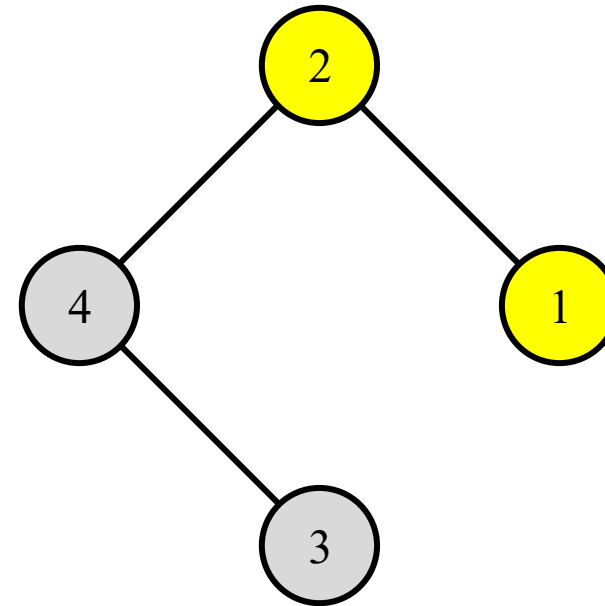
Original node labels



$$\mathbf{H}^0 = (1, 2, 3, 4)^{\top}$$

Relabel

Swap labels 1 and 2



$$P\mathbf{H}^0 = (2, 1, 3, 4)^{\top}$$

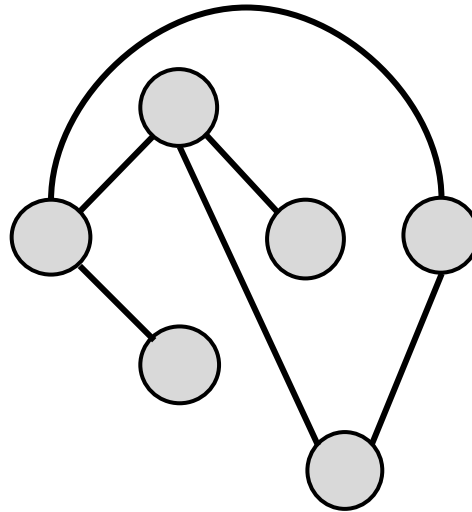
Outline

- Applications of Graphs
- Background
- **Challenges of Deep Learning on Graphs**
- GNNs
 - Overview
 - Message Passing
 - Architectures and Attention
 - Readout
- Implementation
- Relationship w. Transformers

Deep Learning on Graphs

Key Challenges:

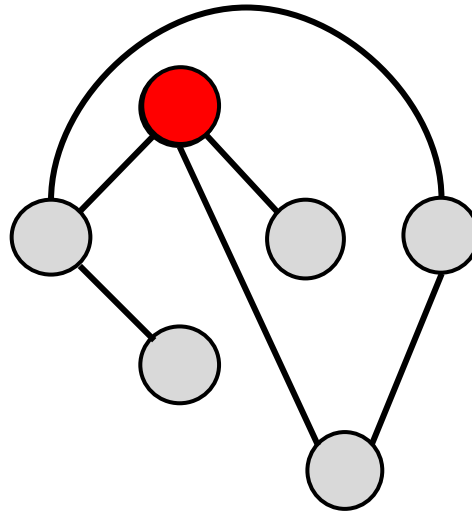
- **Unordered Neighbors**



Deep Learning on Graphs

Key Challenges:

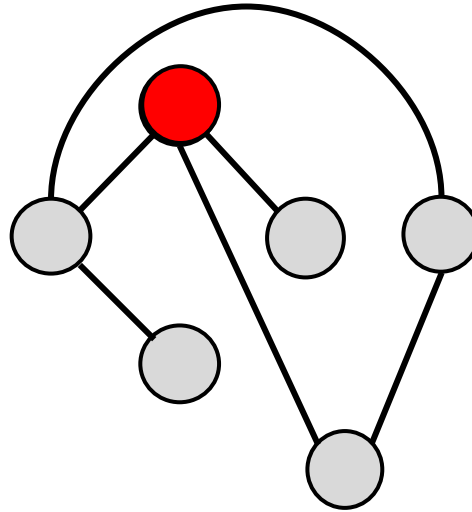
- **Unordered Neighbors**



Deep Learning on Graphs

Key Challenges:

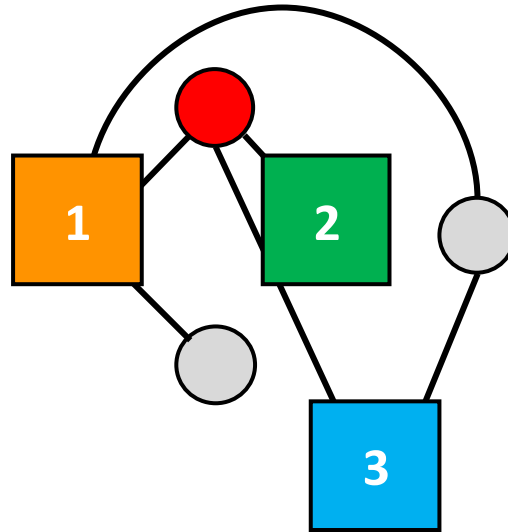
- **Unordered Neighbors**



Deep Learning on Graphs

Key Challenges:

- **Unordered Neighbors**



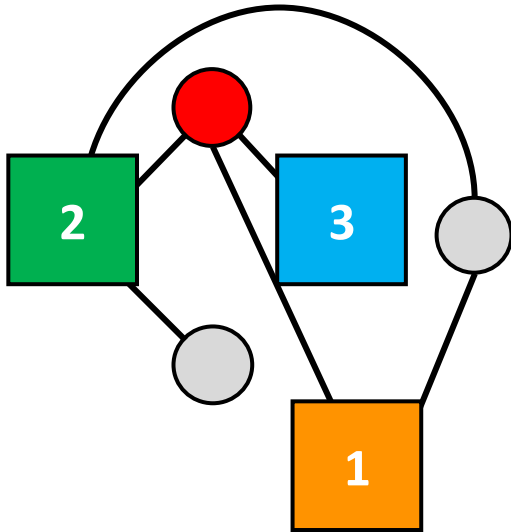
Option 1



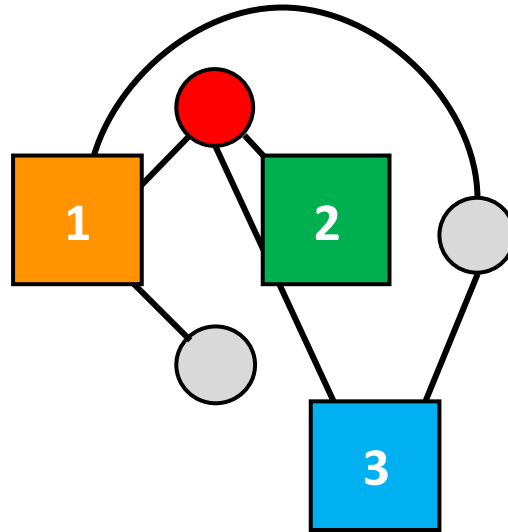
Deep Learning on Graphs

Key Challenges:

- **Unordered Neighbors**



Option 2



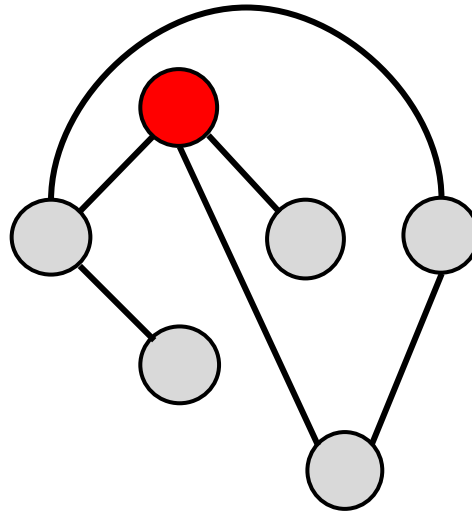
Option 1



Deep Learning on Graphs

Key Challenges:

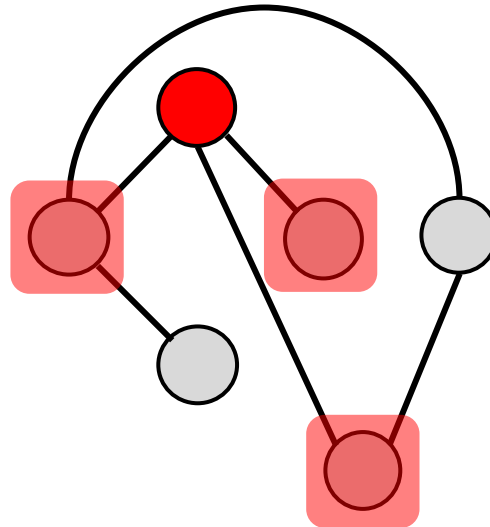
- Unordered Neighbors
- **Varying Neighborhood Sizes**



Deep Learning on Graphs

Key Challenges:

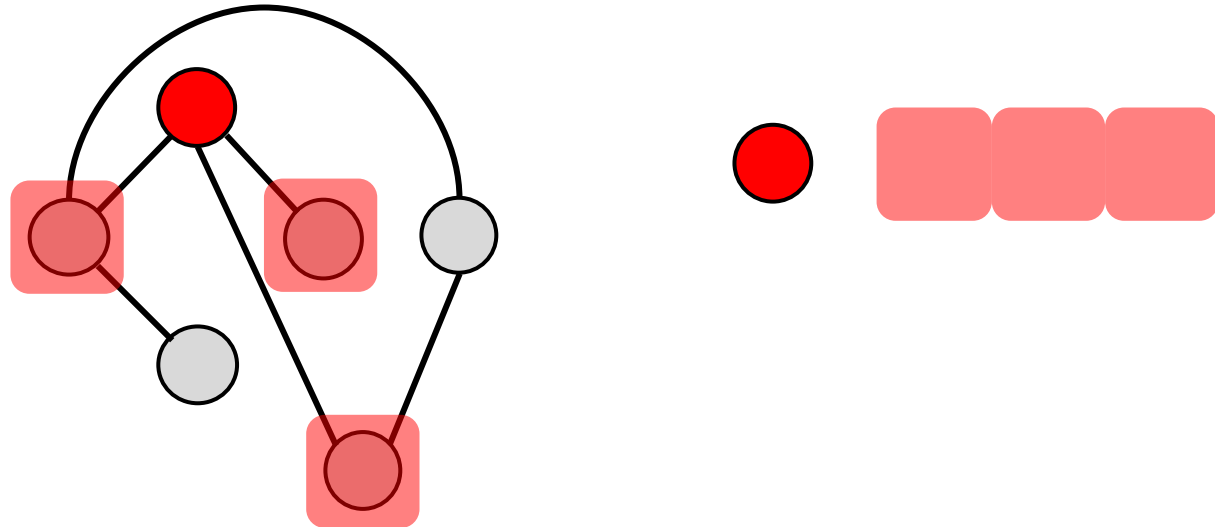
- Unordered Neighbors
- **Varying Neighborhood Sizes**



Deep Learning on Graphs

Key Challenges:

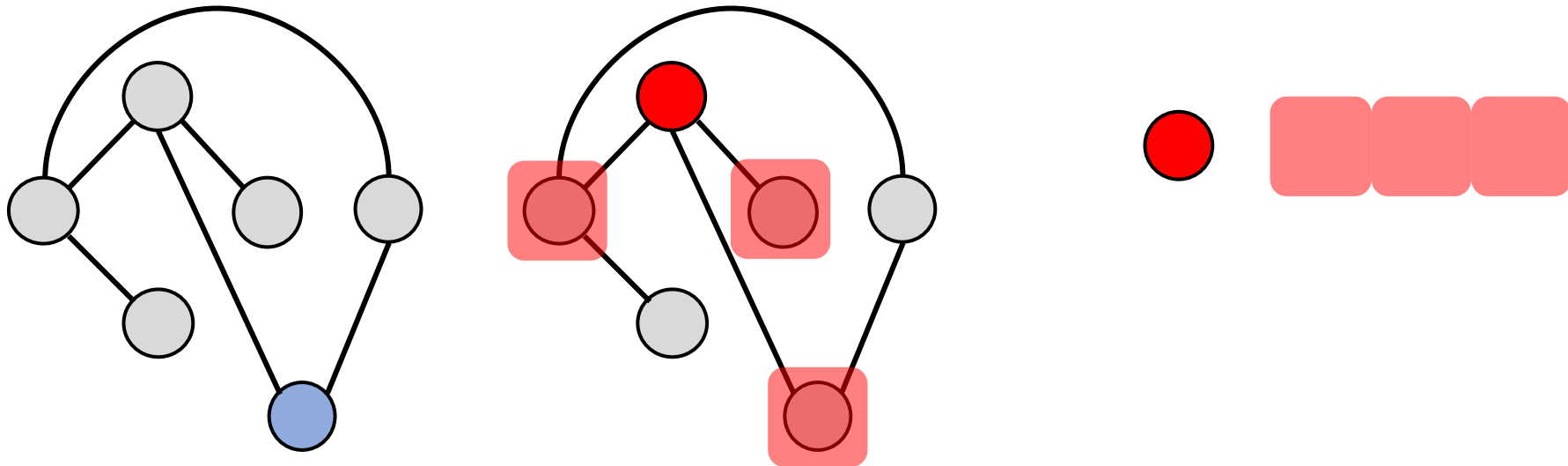
- Unordered Neighbors
- **Varying Neighborhood Sizes**



Deep Learning on Graphs

Key Challenges:

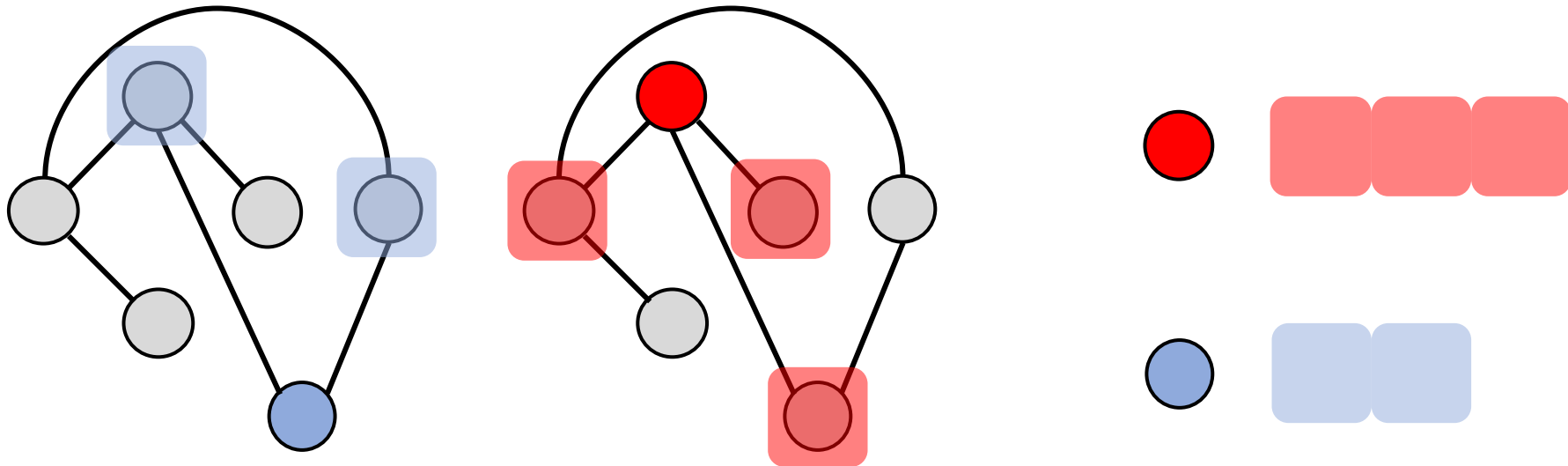
- Unordered Neighbors
- **Varying Neighborhood Sizes**



Deep Learning on Graphs

Key Challenges:

- Unordered Neighbors
- **Varying Neighborhood Sizes**



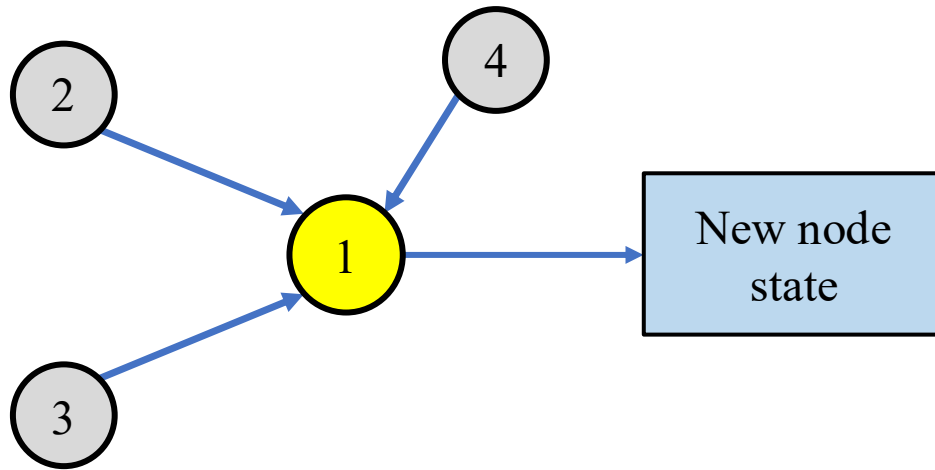
Outline

- Applications of Graphs
- Background
- Challenges of Deep Learning on Graphs
- GNNs
 - **Overview**
 - Message Passing
 - Architectures and Attention
 - Readout
- Implementation
- Relationship w. Transformers

Graph Neural Networks

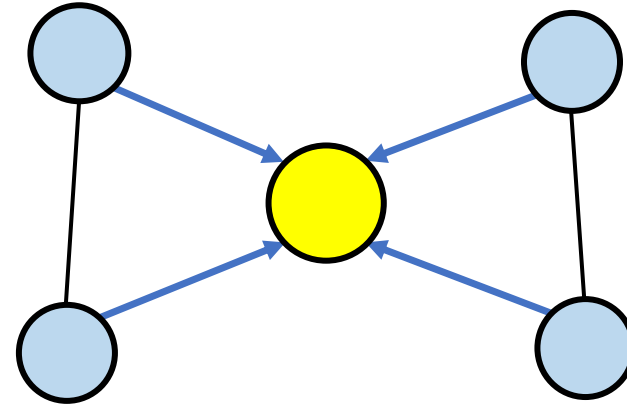
Two broad approaches

Message passing



Exchange messages along edges and update each node state.

Graph convolution

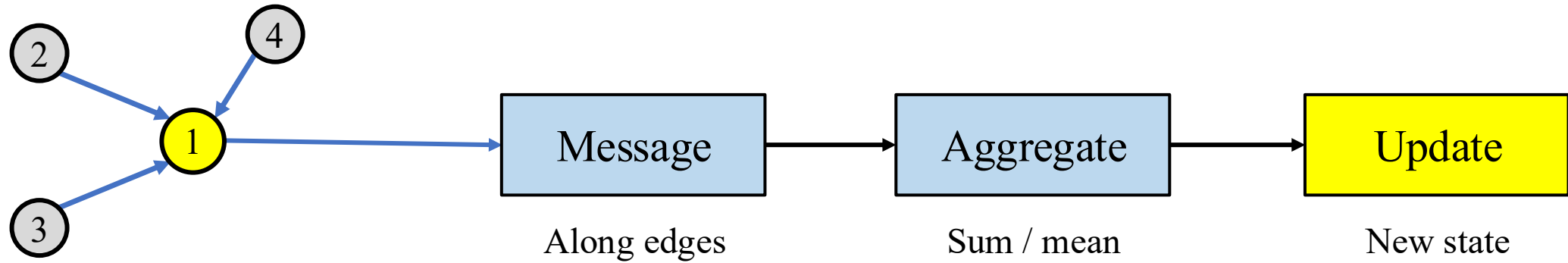


Apply shared filters to signals defined on the graph.

These viewpoints overlap: many graph convolutions are message-passing layers.

Graph Neural Networks

Message-passing approaches



Early GNNs [13, 2]: shared recurrent updates

Recursive nets [3]: structured predecessors

GGNN [9]: gated node updates

MPNN [5]: unifies messages and aggregation

Attention [11]: weights neighboring messages

GraphSAGE [8]: samples node neighborhoods

GPNN [6]: local and inter-part propagation

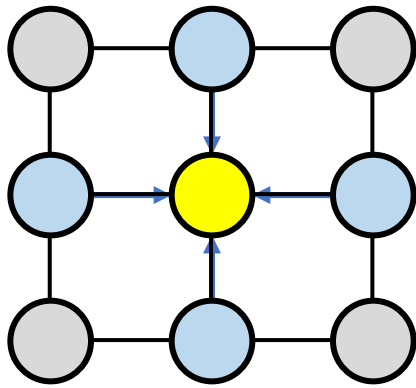
Higher-order [7]: states on node tuples

Boltzmann [4]: related probabilistic models

Graph Neural Networks

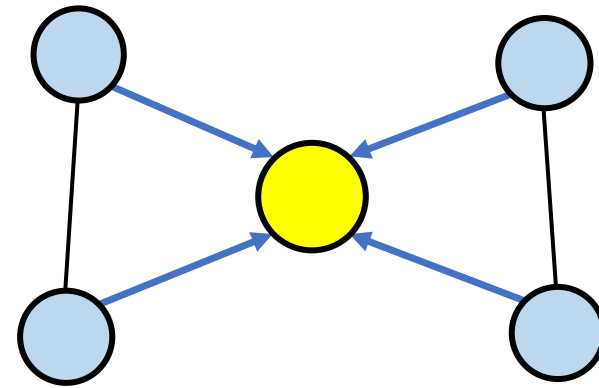
Graph-convolution approaches

Generalize shared filtering from a grid to a graph.



Regular grid

Shared local
filtering



Irregular graph

Bruna et al. [18]

Filters in graph-Laplacian eigenbasis

GCN [20]

Normalized neighbor aggregation

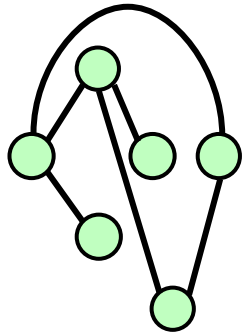
ChebNet [19]

Local polynomial filters of the Laplacian

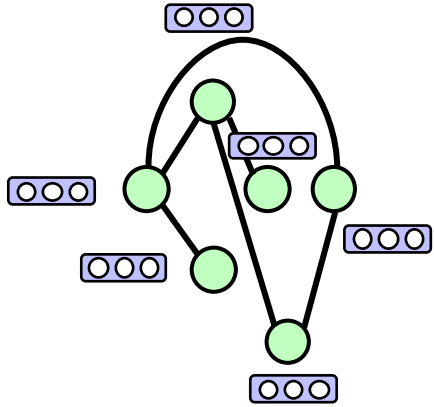
PinSage [15]

Sampled neighbor convolutions for recommendation

Graph Neural Networks (GNNs)

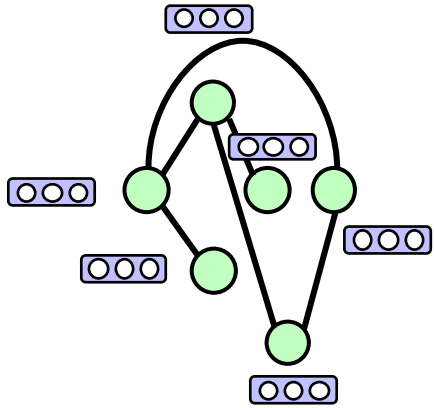


Graph Neural Networks (GNNs)



Input Encoding

Graph Neural Networks (GNNs)



Input Encoding

1. Node Feature

- *Without attributes, use constant or degree features*

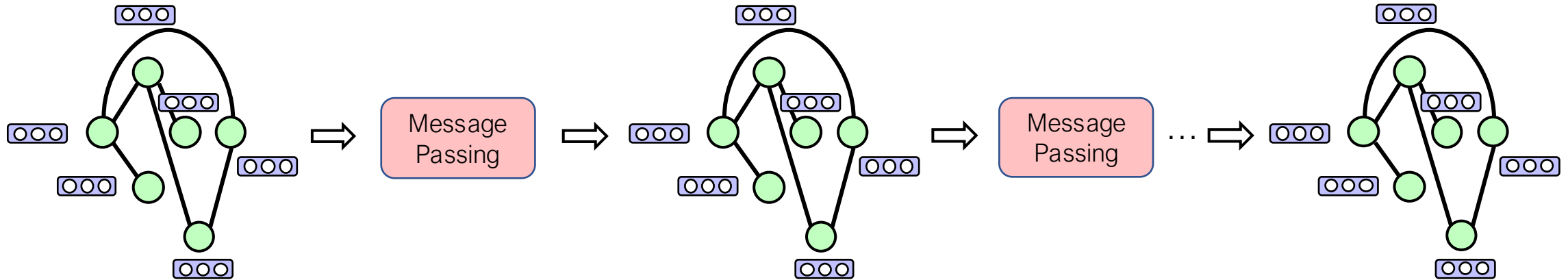
2. Edge Feature

- *Include edge features in the message network*

3. Graph Feature

- *Option: use a virtual node for graph features*
- *Or include graph features in the readout*

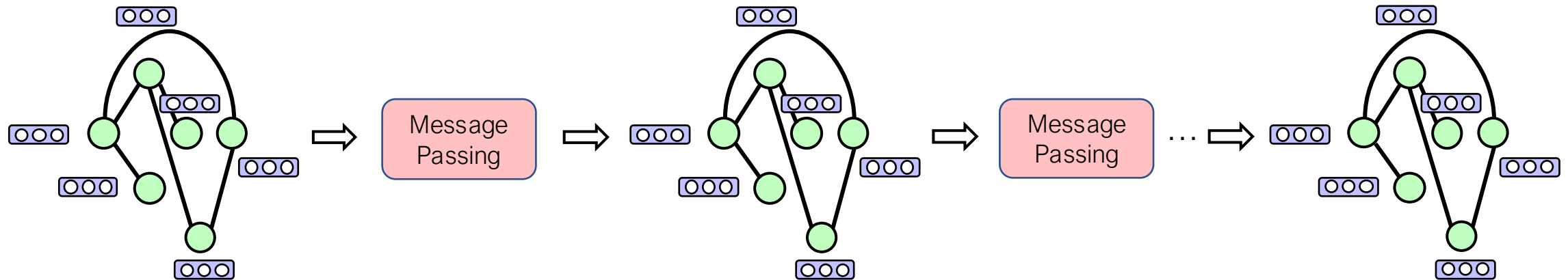
Graph Neural Networks (GNNs)



Input Encoding

Message Passing Layers/Steps

Graph Neural Networks (GNNs)



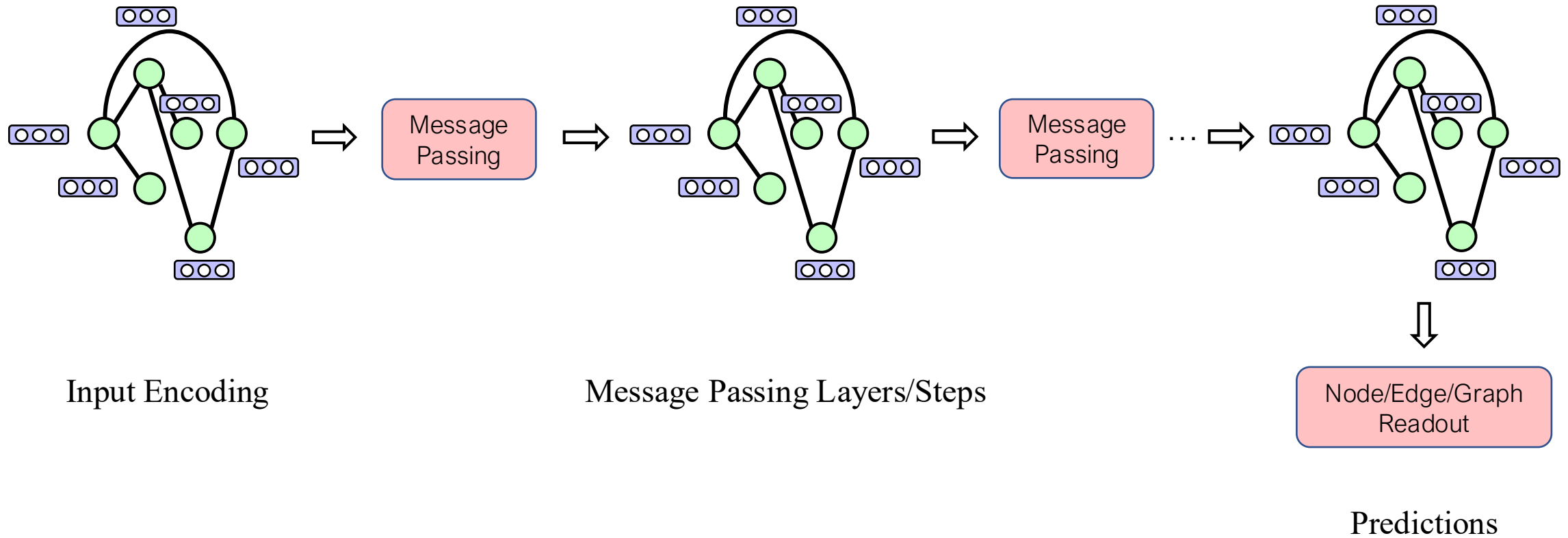
Input Encoding

Message Passing Layers/Steps

Recurrent GNN: share weights across message-passing steps

Feedforward GNN: allow layer-specific weights

Graph Neural Networks (GNNs)

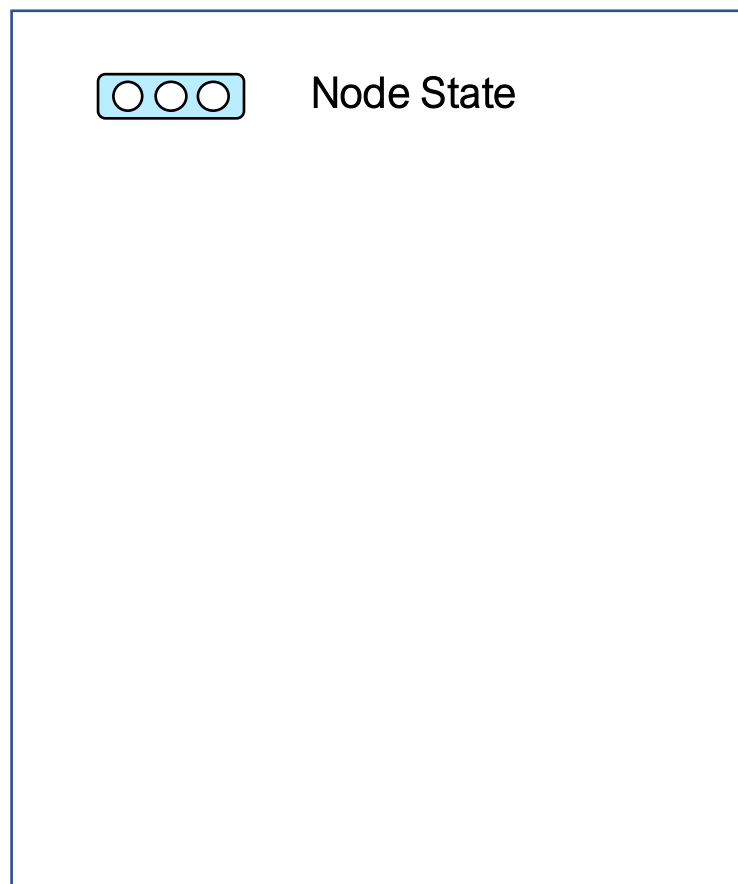


Outline

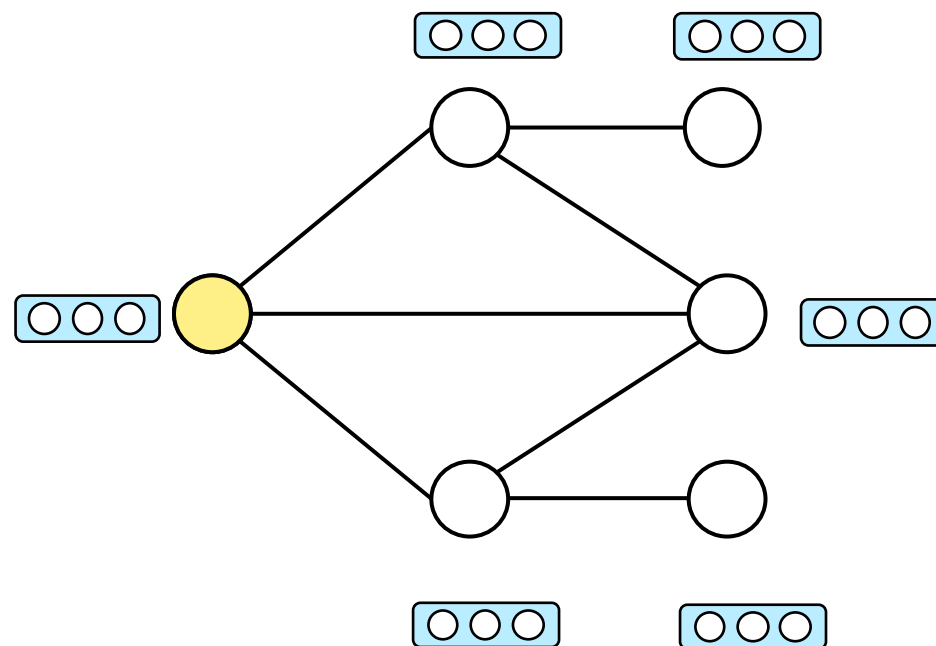
- Applications of Graphs
- Background
- Challenges of Deep Learning on Graphs
- GNNs
 - Overview
 - **Message Passing**
 - Architectures and Attention
 - Readout
- Implementation
- Relationship w. Transformers

Message Passing in GNNs

$\mathbf{h}_i^t$

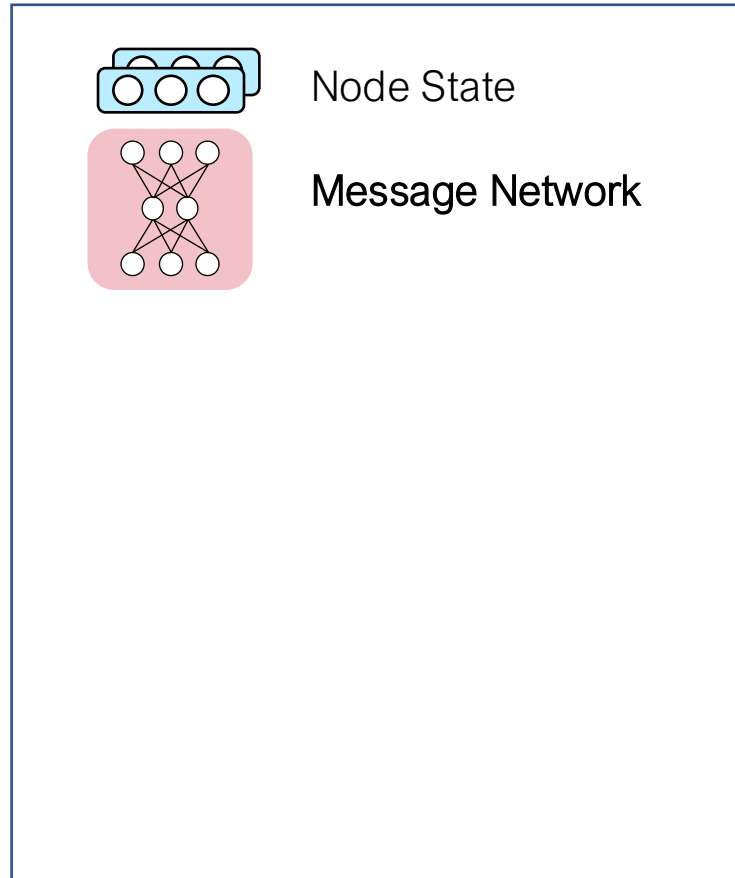


(t+1)-th message passing step/layer

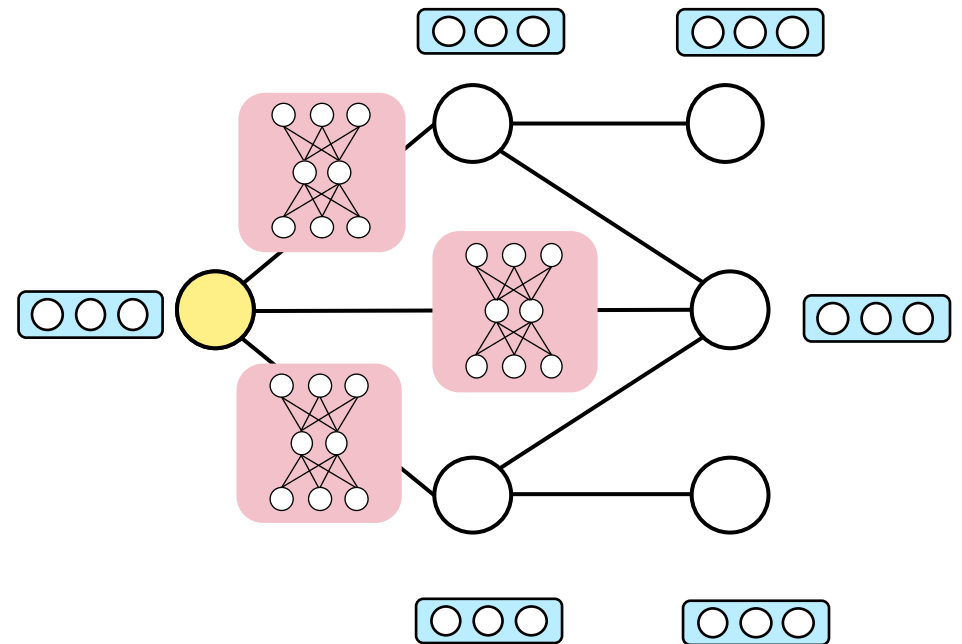


Message Passing in GNNs

$\mathbf{h}_i^t$ $\mathbf{h}_j^t$



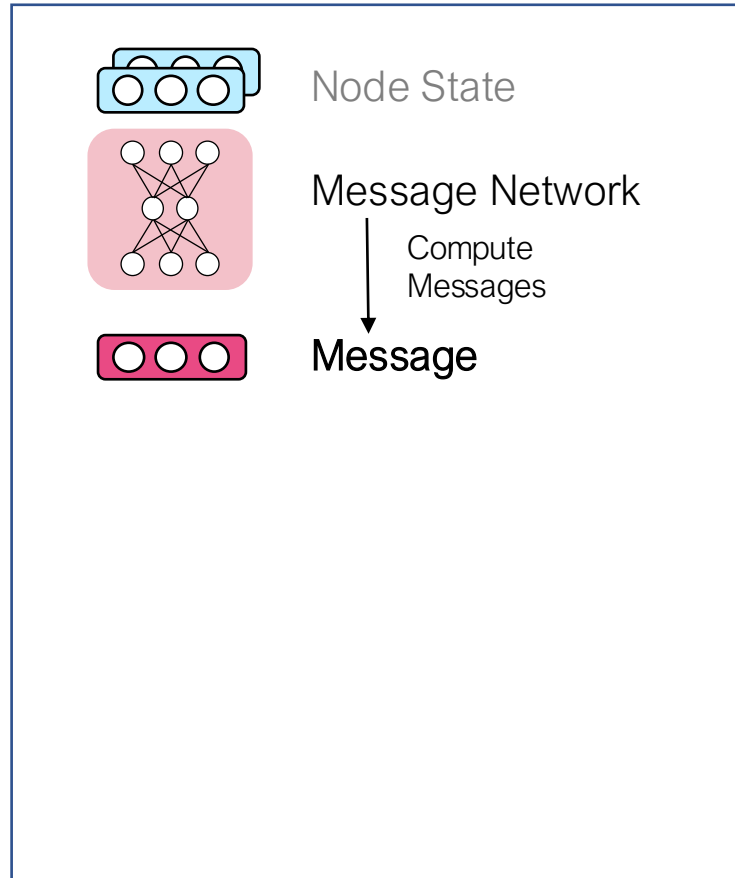
(t+1)-th message passing step/layer



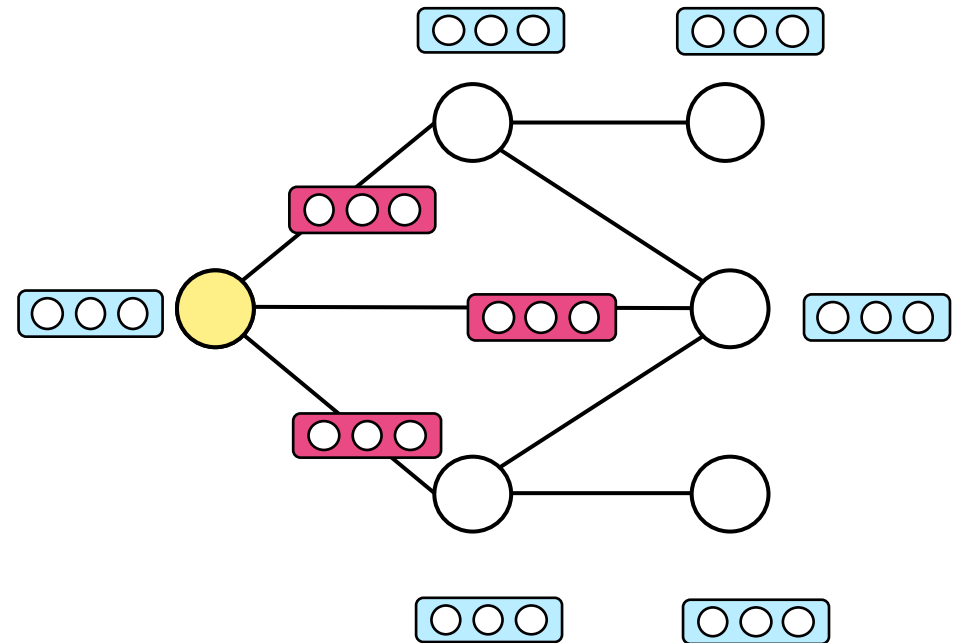
Message Passing in GNNs

$\mathbf{h}_i^t$ $\mathbf{h}_j^t$

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

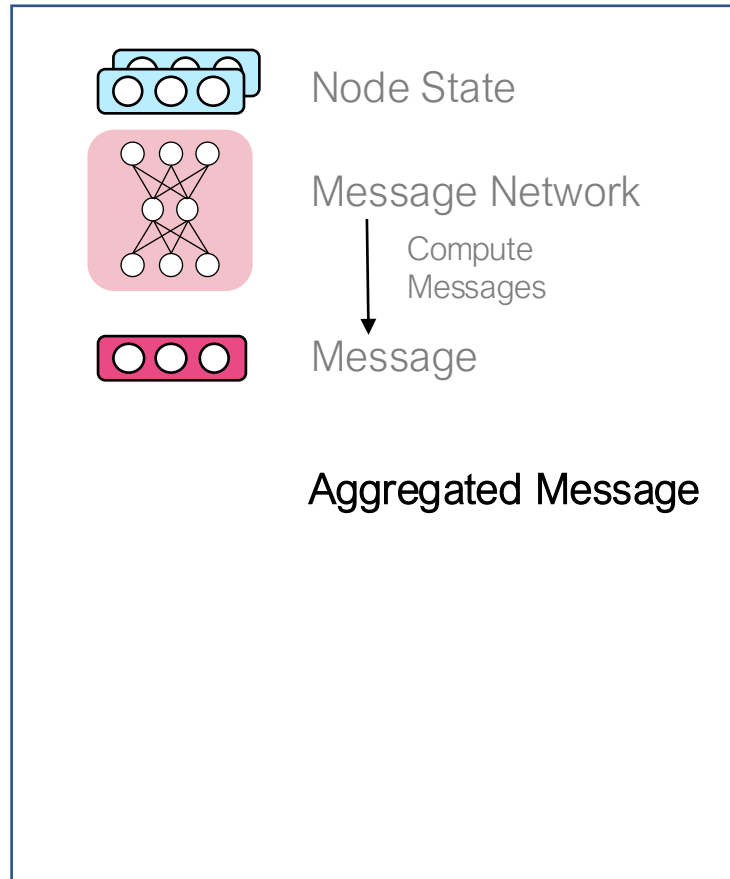


(t+1)-th message passing step/layer

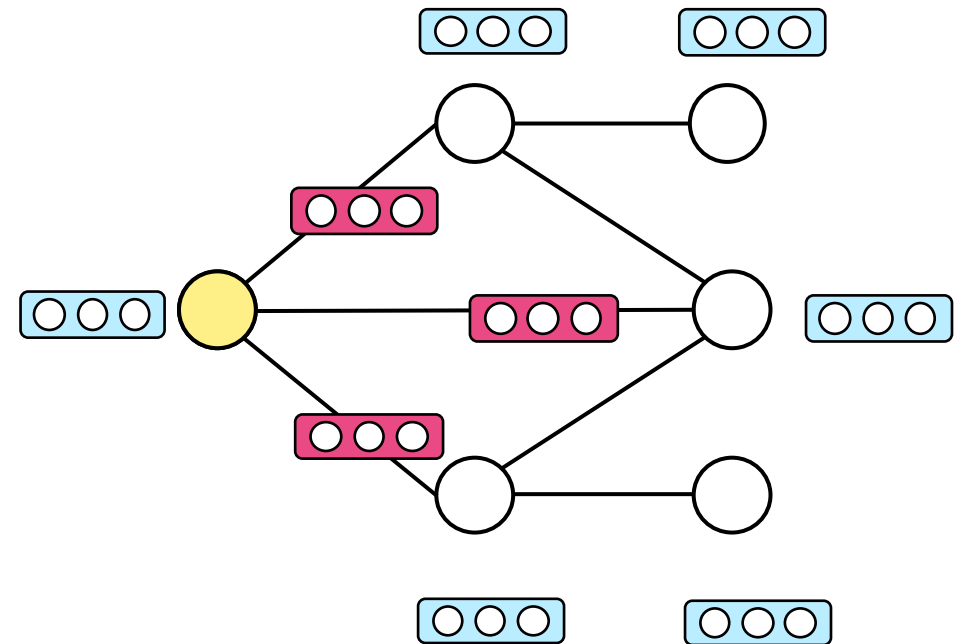


Message Passing in GNNs

$$\mathbf{h}_i^t \quad \mathbf{h}_j^t$$
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$



(t+1)-th message passing step/layer

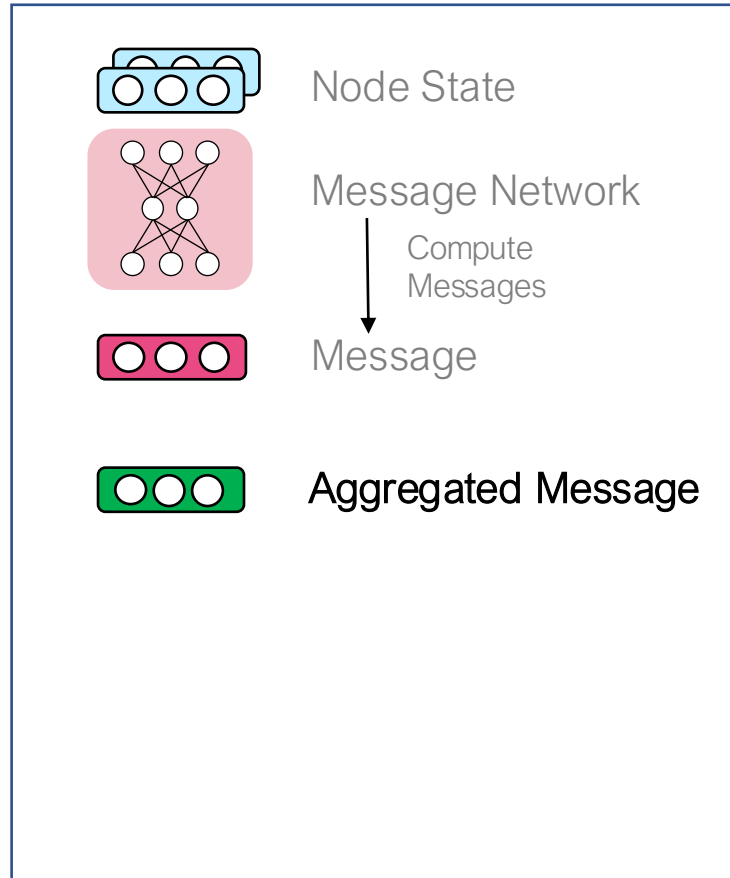


Message Passing in GNNs

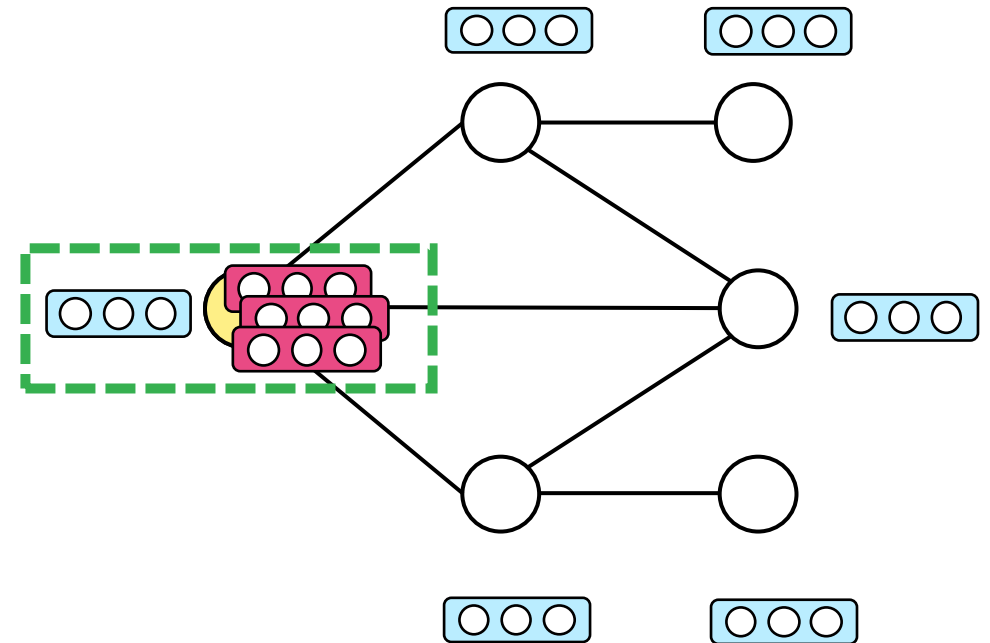
$\mathbf{h}_i^t \quad \mathbf{h}_j^t$

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$



(t+1)-th message passing step/layer

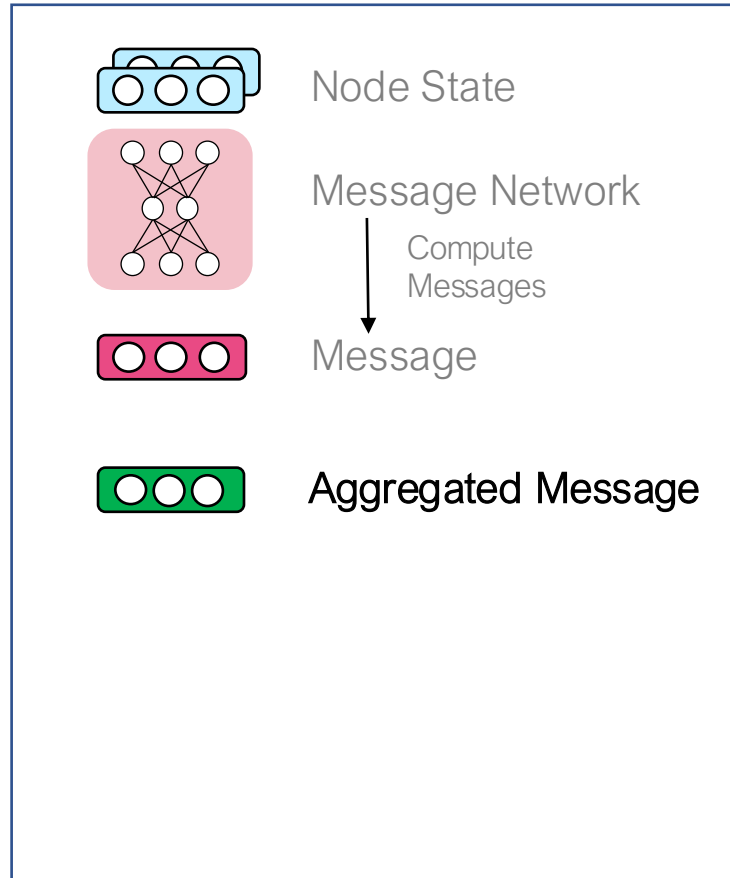


Message Passing in GNNs

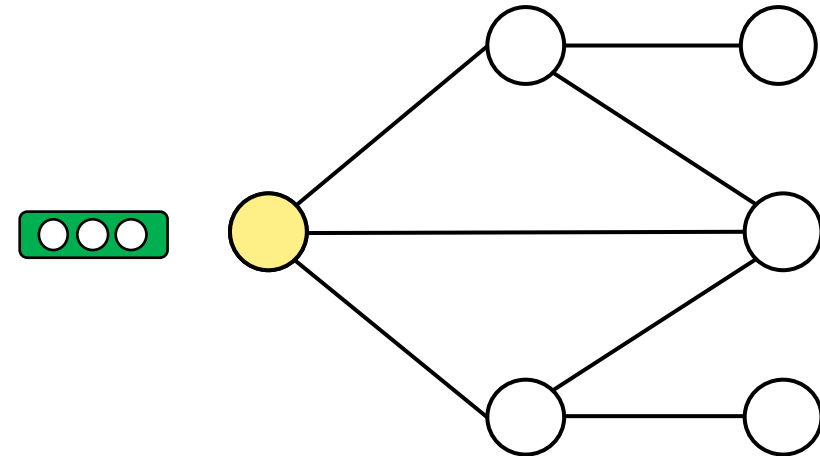
$$\mathbf{h}_i^t \quad \mathbf{h}_j^t$$

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$



(t+1)-th message passing step/layer

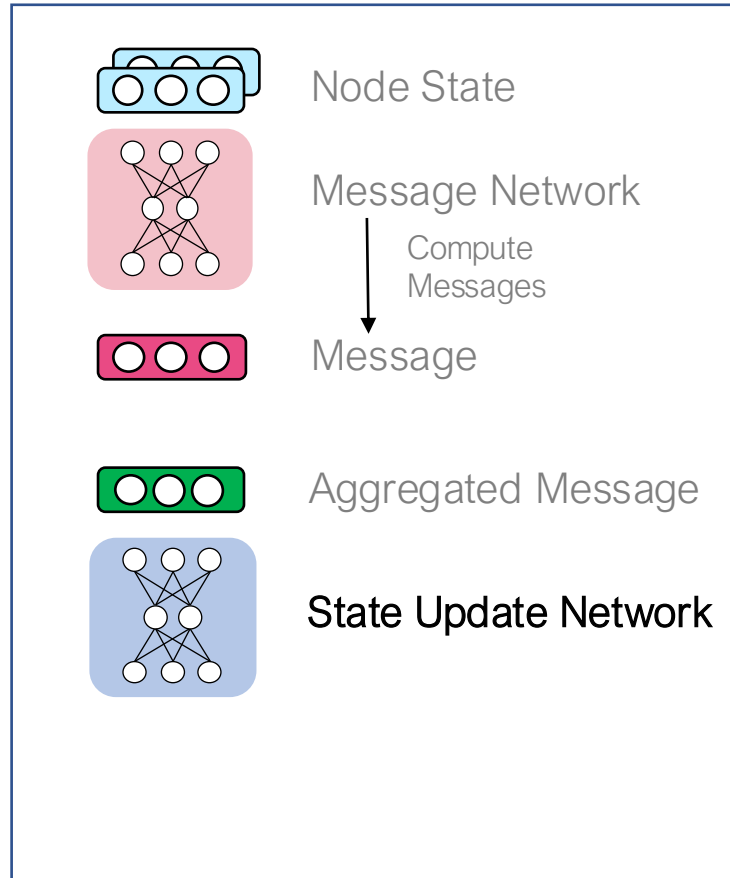


Message Passing in GNNs

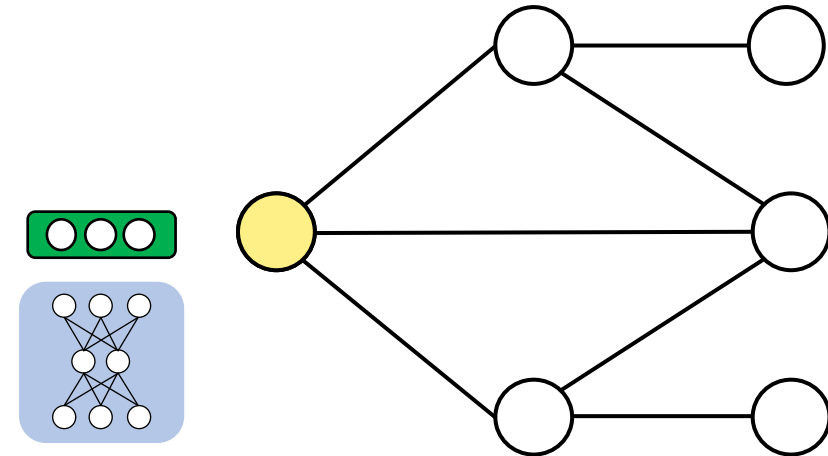
$$\mathbf{h}_i^t \quad \mathbf{h}_j^t$$

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$



(t+1)-th message passing step/layer



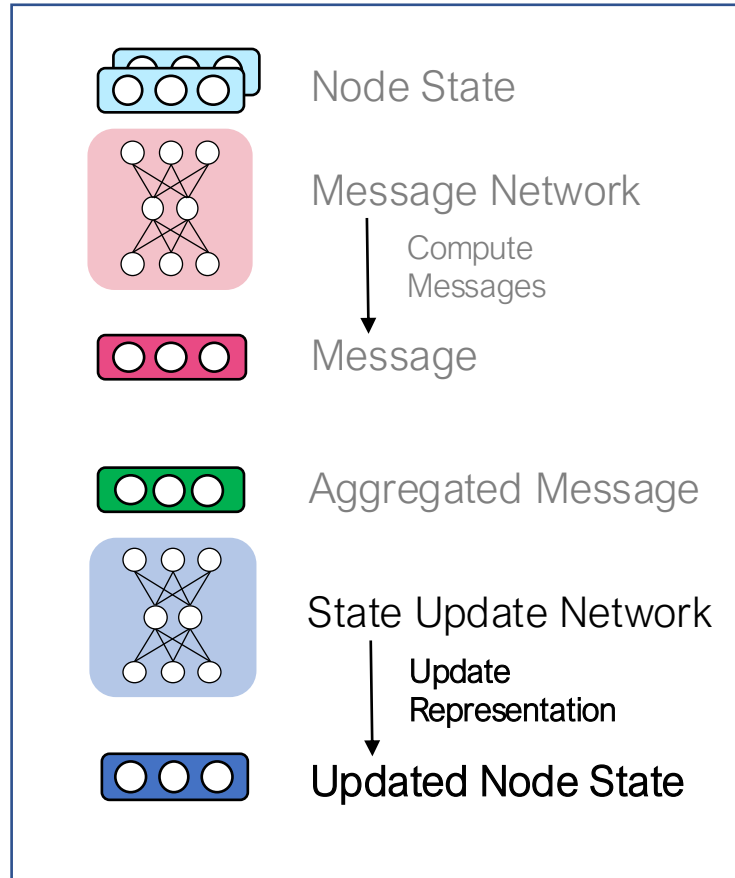
Message Passing in GNNs

$$\mathbf{h}_i^t \quad \mathbf{h}_j^t$$

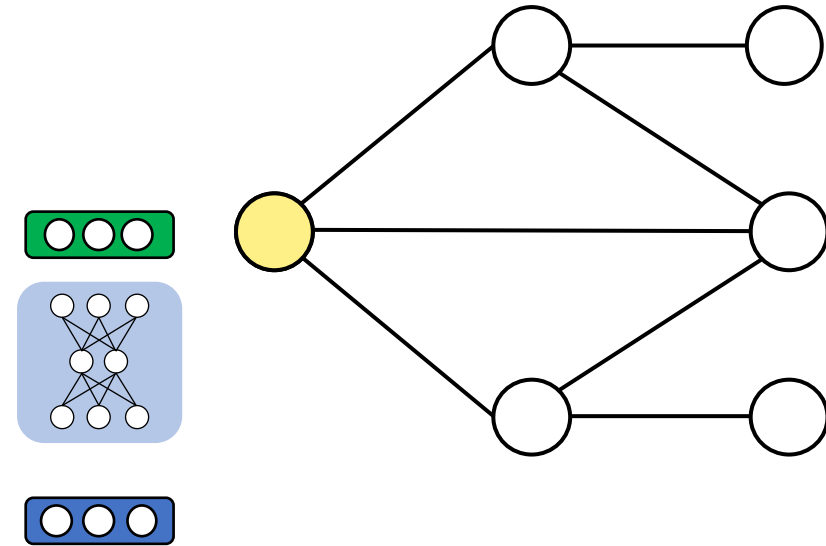
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

$$\mathbf{h}_i^{t+1} = f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$



(t+1)-th message passing step/layer



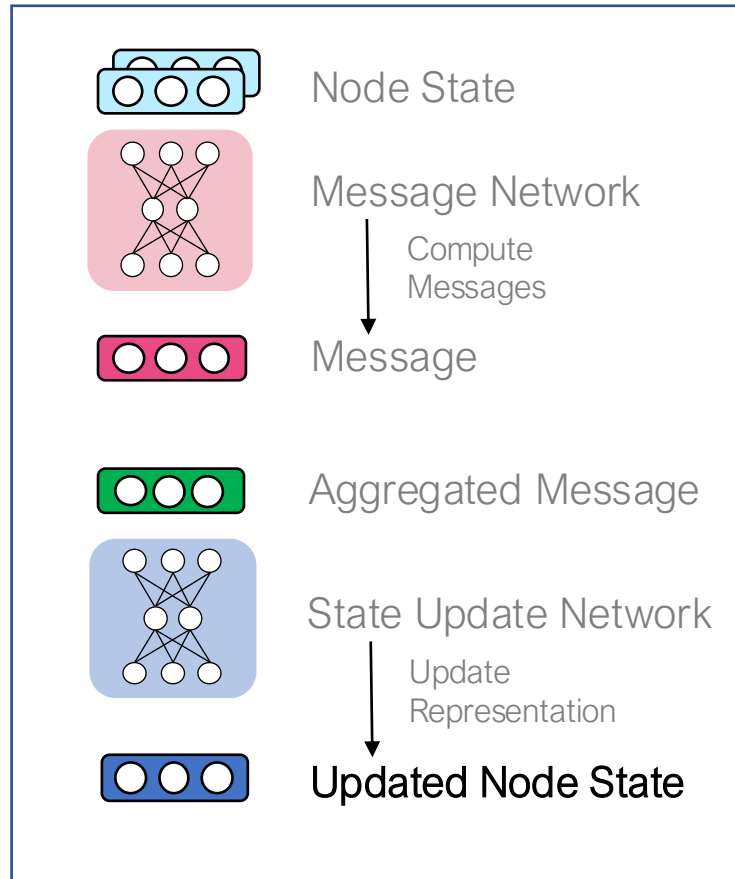
Message Passing in GNNs

$$\mathbf{h}_i^t \quad \mathbf{h}_j^t$$

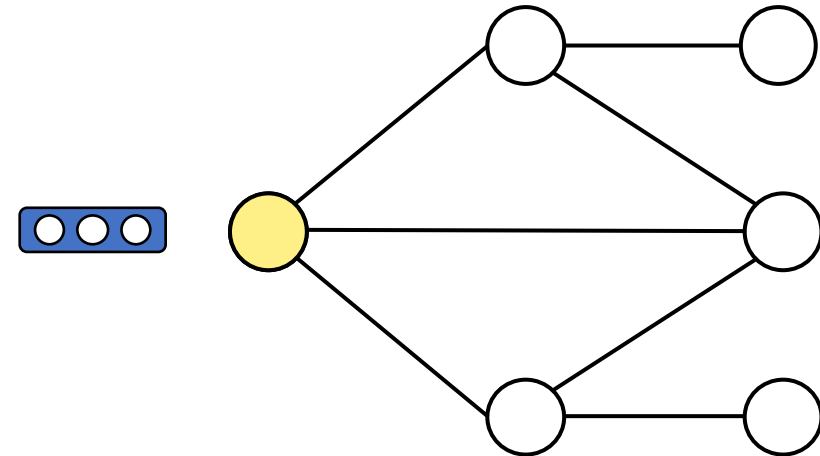
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

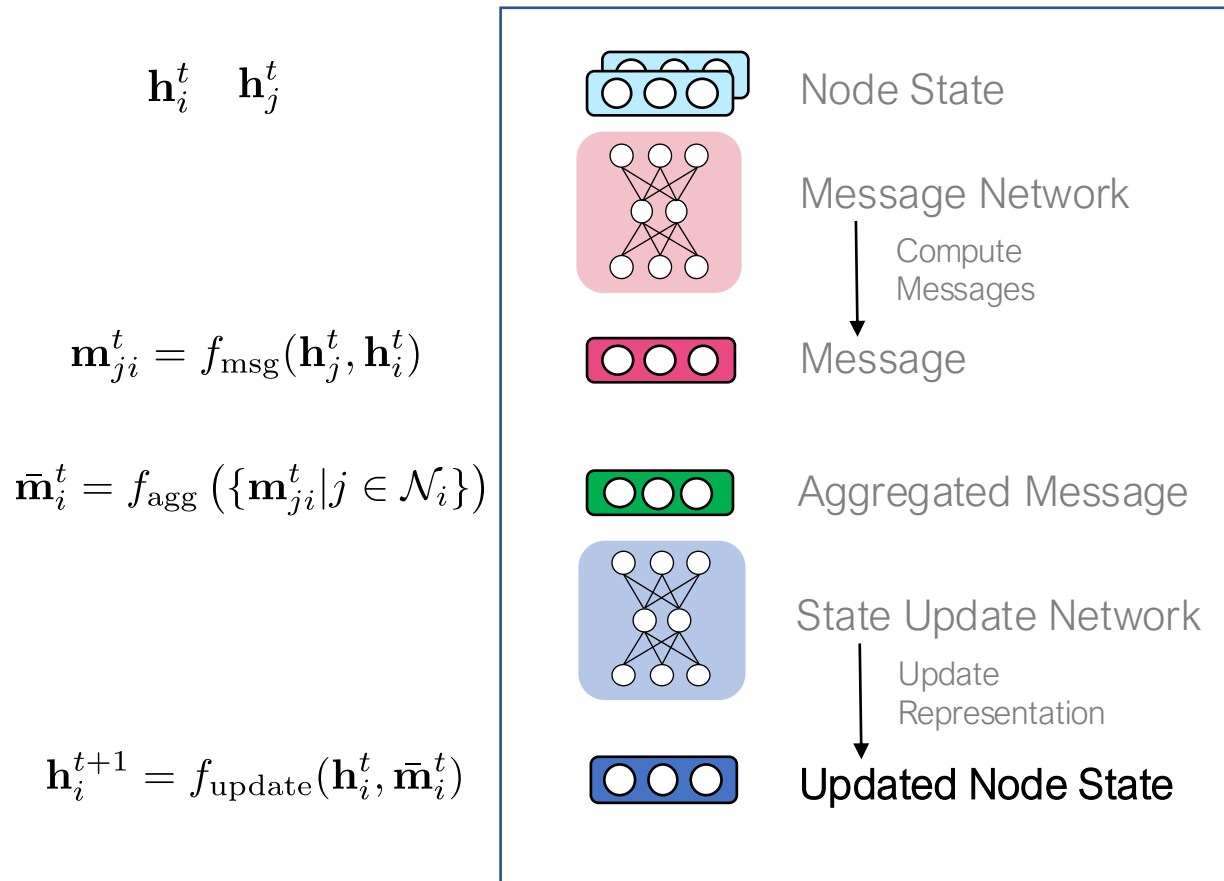
$$\mathbf{h}_i^{t+1} = f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$



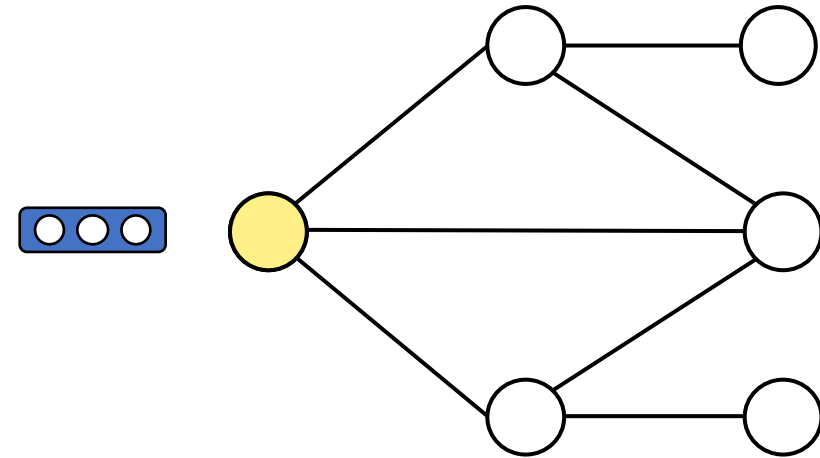
(t+1)-th message passing step/layer



Message Passing in GNNs



(t+1)-th message passing step/layer



- **Parallel Schedule!**

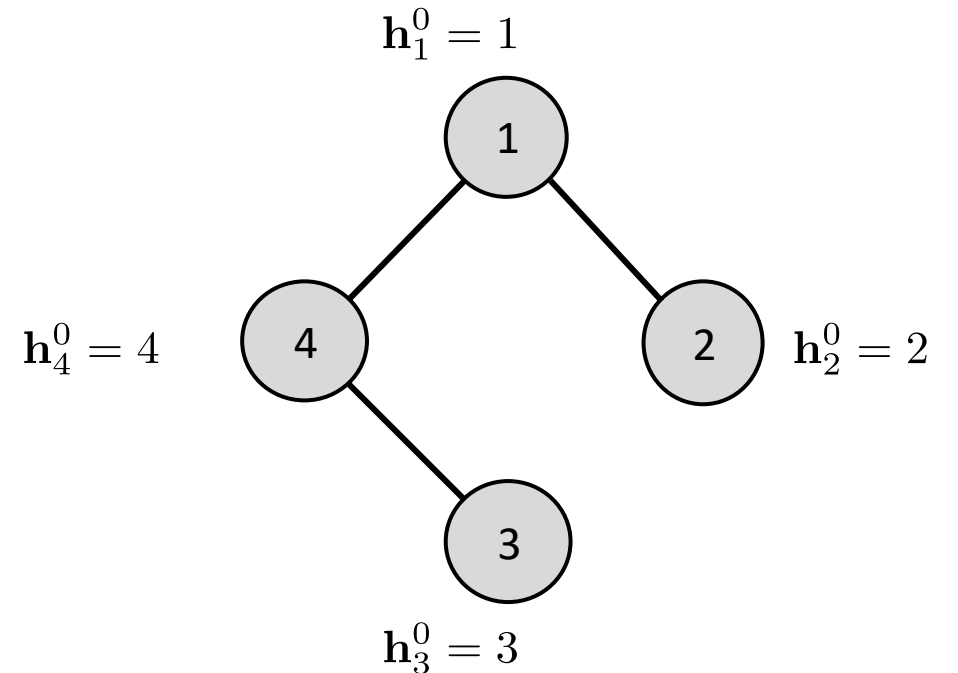
Message Passing: A Worked Example

One scalar feature per node. Use the same rule at every node.

Neighbors exclude the node itself; all updates use the old states.

$$\mathbf{H}^0 = (1, 2, 3, 4)^\top$$

$$\mathbf{h}_i^{t+1} = \text{ReLU} \left(\mathbf{h}_i^t + \sum_{j \in \mathcal{N}_i} \mathbf{h}_j^t \right)$$



This illustrative update has no learned parameters.

Message Passing: A Worked Example

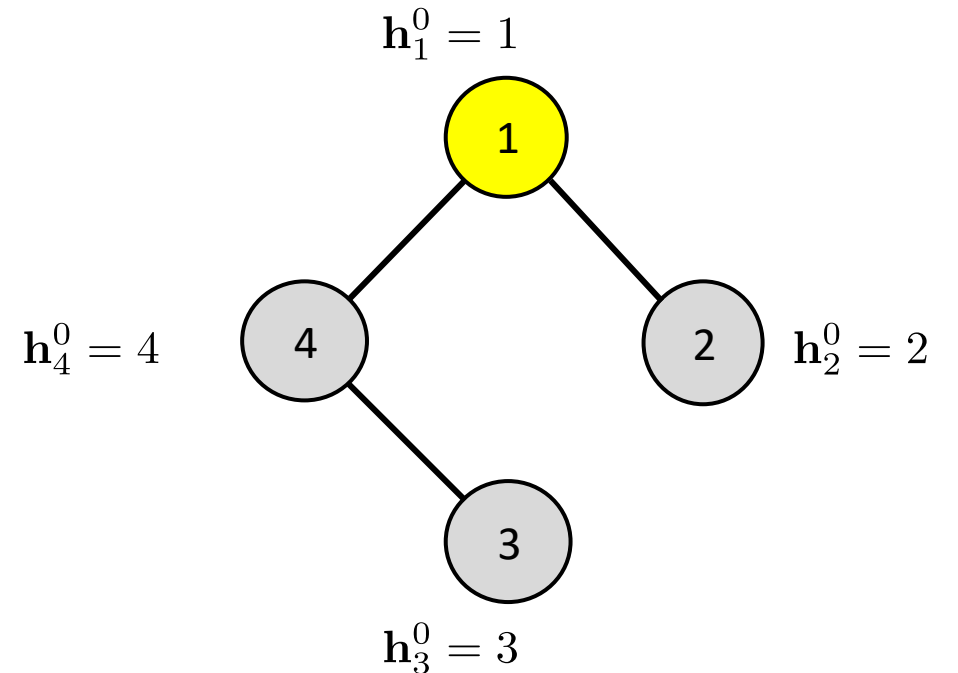
One scalar feature per node. Use the same rule at every node.

Neighbors exclude the node itself; all updates use the old states.

$$\mathbf{H}^0 = (1, 2, 3, 4)^\top$$

$$\mathbf{h}_i^{t+1} = \text{ReLU} \left(\mathbf{h}_i^t + \sum_{j \in \mathcal{N}_i} \mathbf{h}_j^t \right)$$

$$\bar{\mathbf{m}}_1^0 = 2 + 4 = 6, \quad \mathbf{h}_1^1 = \text{ReLU}(1 + 6) = 7$$



Message Passing: A Worked Example

One scalar feature per node. Use the same rule at every node.

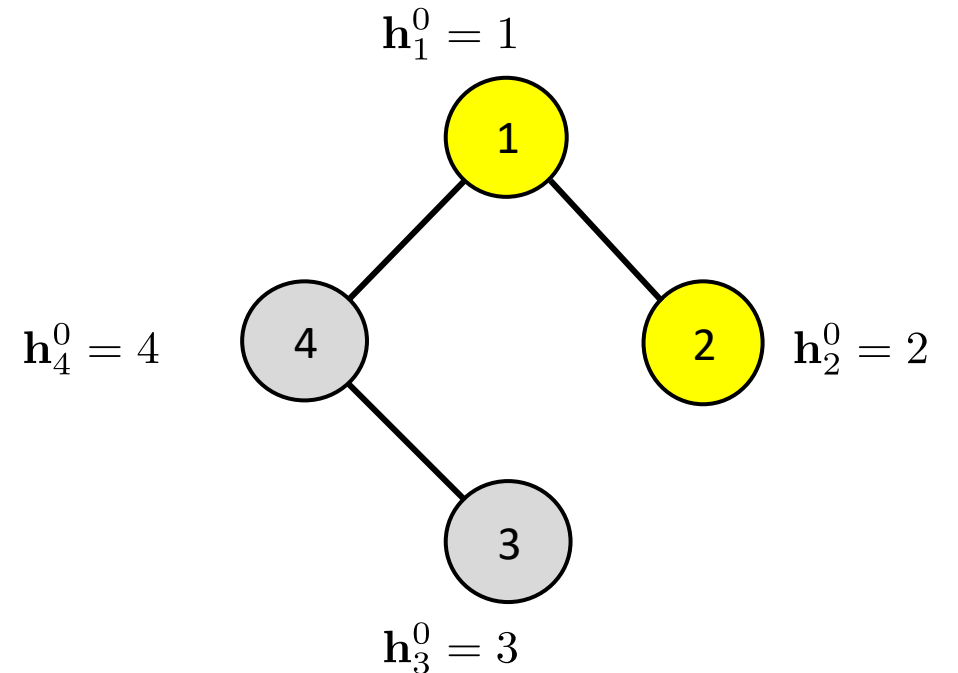
Neighbors exclude the node itself; all updates use the old states.

$$\mathbf{H}^0 = (1, 2, 3, 4)^\top$$

$$\mathbf{h}_i^{t+1} = \text{ReLU} \left(\mathbf{h}_i^t + \sum_{j \in \mathcal{N}_i} \mathbf{h}_j^t \right)$$

$$\bar{\mathbf{m}}_1^0 = 2 + 4 = 6, \quad \mathbf{h}_1^1 = \text{ReLU}(1 + 6) = 7$$

$$\mathbf{h}_2^1 = \text{ReLU}(2 + 1) = 3$$



Message Passing: A Worked Example

One scalar feature per node. Use the same rule at every node.

Neighbors exclude the node itself; all updates use the old states.

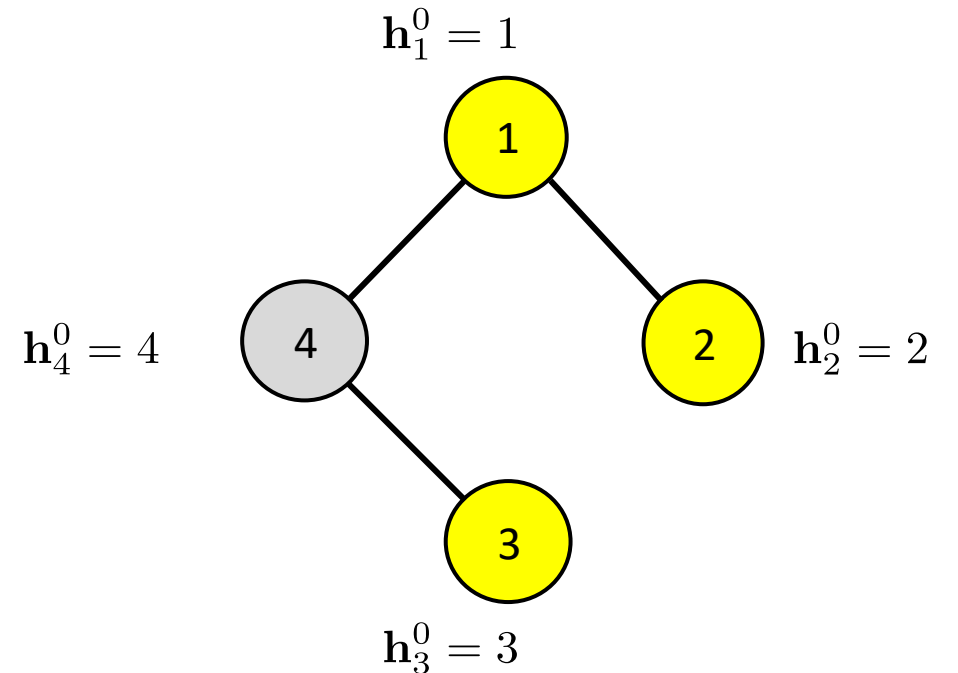
$$\mathbf{H}^0 = (1, 2, 3, 4)^\top$$

$$\mathbf{h}_i^{t+1} = \text{ReLU} \left(\mathbf{h}_i^t + \sum_{j \in \mathcal{N}_i} \mathbf{h}_j^t \right)$$

$$\bar{\mathbf{m}}_1^0 = 2 + 4 = 6, \quad \mathbf{h}_1^1 = \text{ReLU}(1 + 6) = 7$$

$$\mathbf{h}_2^1 = \text{ReLU}(2 + 1) = 3$$

$$\mathbf{h}_3^1 = \text{ReLU}(3 + 4) = 7$$



Message Passing: A Worked Example

One scalar feature per node. Use the same rule at every node.

Neighbors exclude the node itself; all updates use the old states.

$$\mathbf{H}^0 = (1, 2, 3, 4)^\top$$

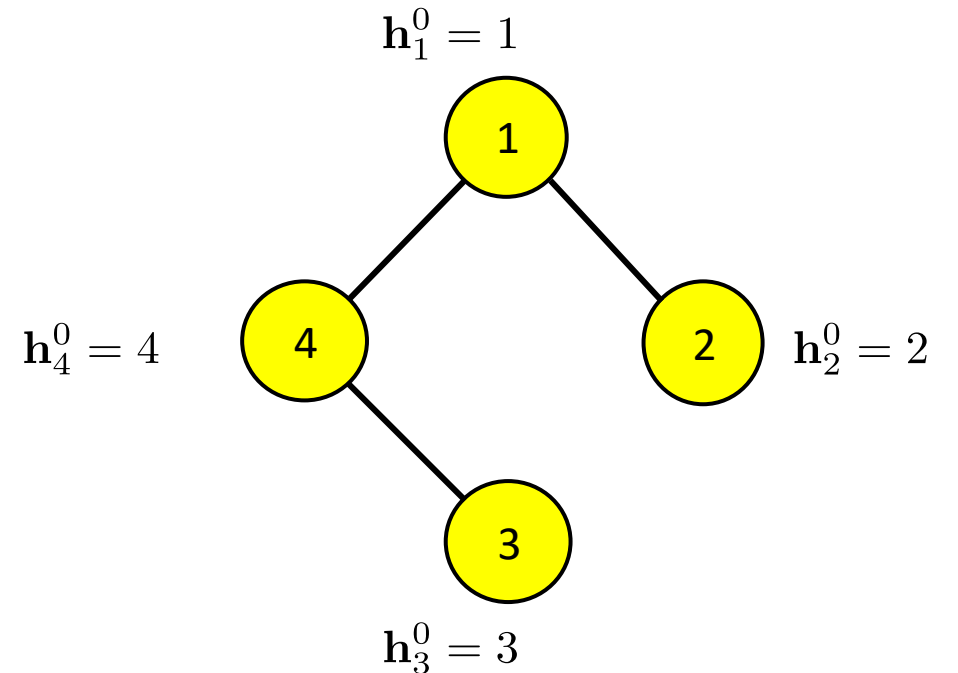
$$\mathbf{h}_i^{t+1} = \text{ReLU} \left(\mathbf{h}_i^t + \sum_{j \in \mathcal{N}_i} \mathbf{h}_j^t \right)$$

$$\bar{\mathbf{m}}_1^0 = 2 + 4 = 6, \quad \mathbf{h}_1^1 = \text{ReLU}(1 + 6) = 7$$

$$\mathbf{h}_2^1 = \text{ReLU}(2 + 1) = 3$$

$$\mathbf{h}_3^1 = \text{ReLU}(3 + 4) = 7$$

$$\mathbf{h}_4^1 = \text{ReLU}(4 + 1 + 3) = 8$$



Message Passing: A Worked Example

One scalar feature per node. Use the same rule at every node.

Neighbors exclude the node itself; all updates use the old states.

$$\mathbf{H}^0 = (1, 2, 3, 4)^\top$$

$$\mathbf{h}_i^{t+1} = \text{ReLU} \left(\mathbf{h}_i^t + \sum_{j \in \mathcal{N}_i} \mathbf{h}_j^t \right)$$

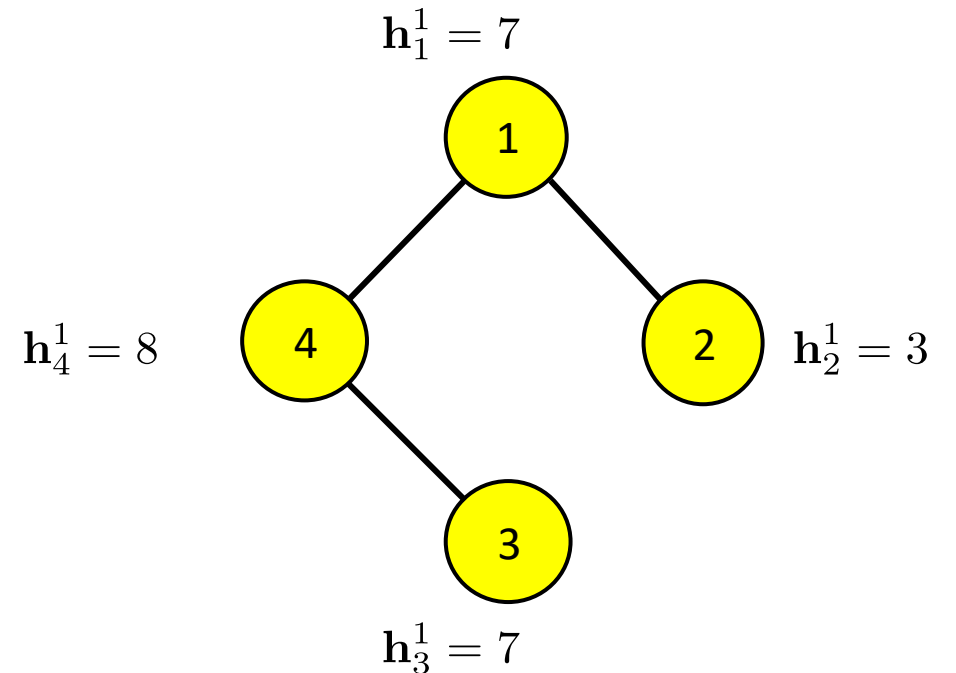
$$\bar{\mathbf{m}}_1^0 = 2 + 4 = 6, \quad \mathbf{h}_1^1 = \text{ReLU}(1 + 6) = 7$$

$$\mathbf{h}_2^1 = \text{ReLU}(2 + 1) = 3$$

$$\mathbf{h}_3^1 = \text{ReLU}(3 + 4) = 7$$

$$\mathbf{h}_4^1 = \text{ReLU}(4 + 1 + 3) = 8$$

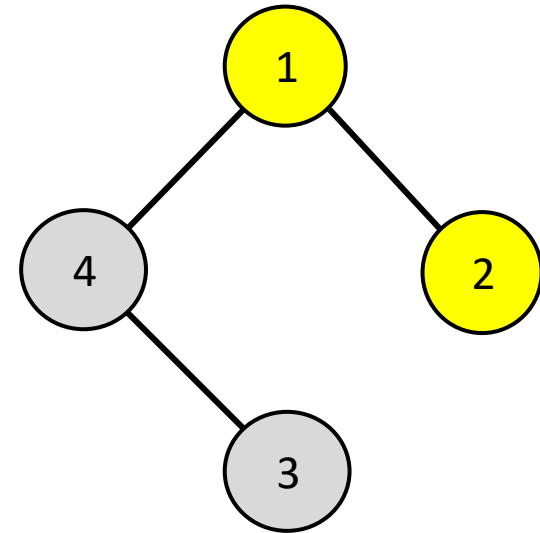
$$\mathbf{H}^1 = (7, 3, 7, 8)^\top$$



Message Passing: Receptive Fields

One layer can use features from one-hop neighbors.

For node 2, this includes nodes 2 and 1.

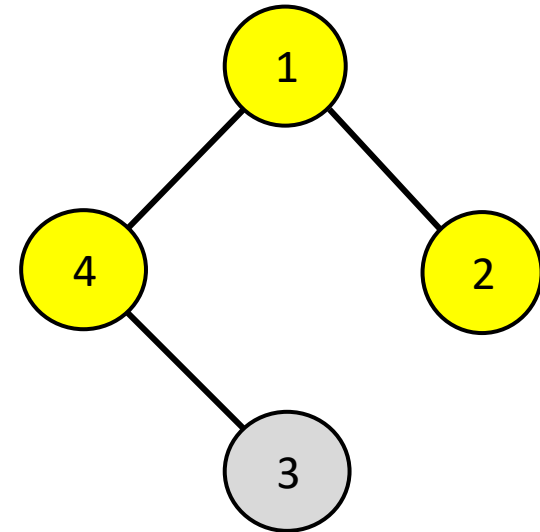


Message Passing: Receptive Fields

One layer can use features from one-hop neighbors.

Two layers can use features from up to two hops away.

Node 3 is three hops from node 2.



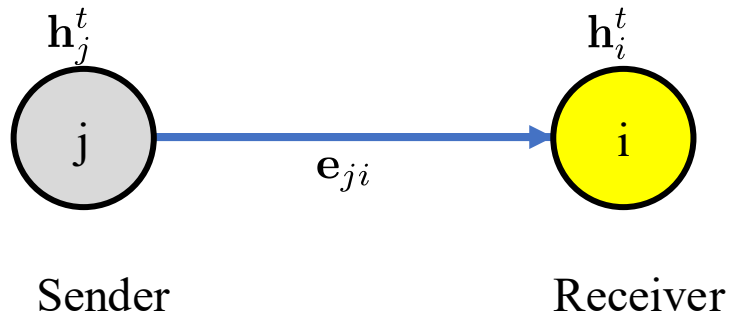
Outline

- Applications of Graphs
- Background
- Challenges of Deep Learning on Graphs
- GNNs
 - Overview
 - Message Passing
 - **Architectures and Attention**
 - Readout
- Implementation
- Relationship w. Transformers

Message Passing: Architecture Choices



The message can copy, transform, or combine node and edge features [5].



e_{ji} : edge feature

Copy the sender state

$$m_{ji}^t = h_j^t$$

Learn a sender projection

$$m_{ji}^t = W_t h_j^t$$

Use the edge feature too

$$m_{ji}^t = \text{MLP}_t([h_j^t, h_i^t, e_{ji}])$$

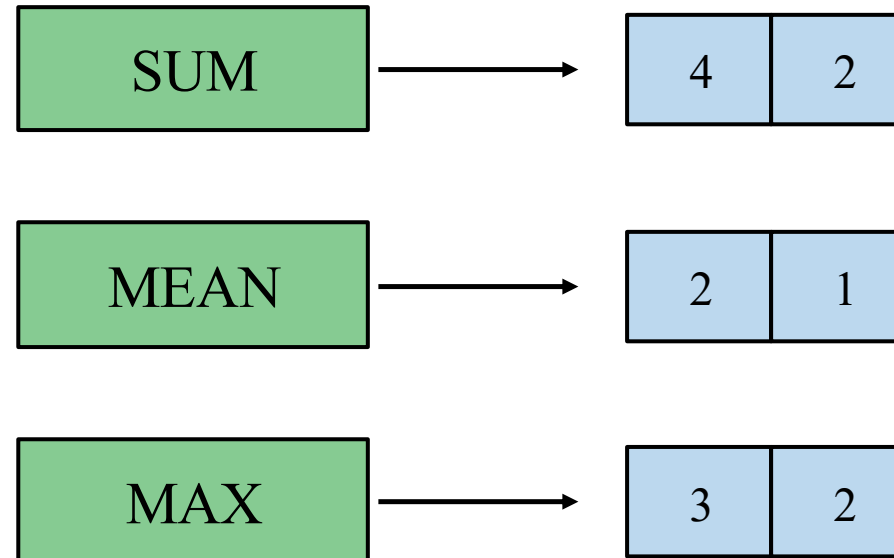
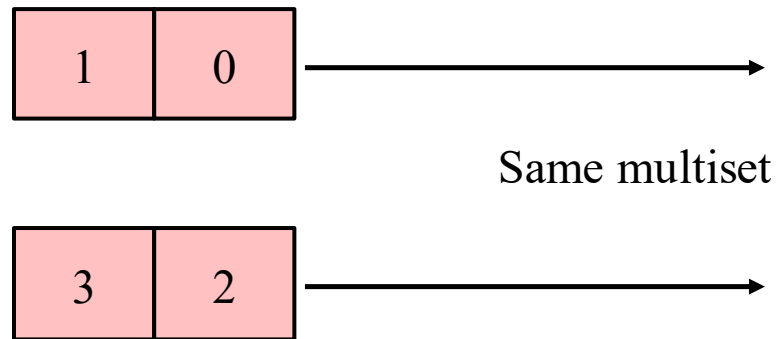
Use the same message function on every edge.

Message Passing: Architecture Choices



Aggregation combines a variable-size, unordered set of messages [5, 8].

Incoming messages



SUM adds messages; MEAN averages; MAX takes a coordinatewise maximum.

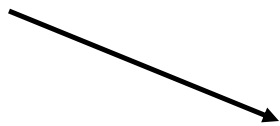
Message Passing: Architecture Choices



The update combines the old node state with the aggregated message.

Old state

$\mathbf{h}_i^t$



$\bar{\mathbf{m}}_i^t$



Neighbor summary



MLP on concatenated inputs [8]

$$\mathbf{h}_i^{t+1} = \text{MLP}_t([\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t])$$

Gated recurrent update [5, 9]

$$\mathbf{h}_i^{t+1} = \text{GRU}_t(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$

Residual update

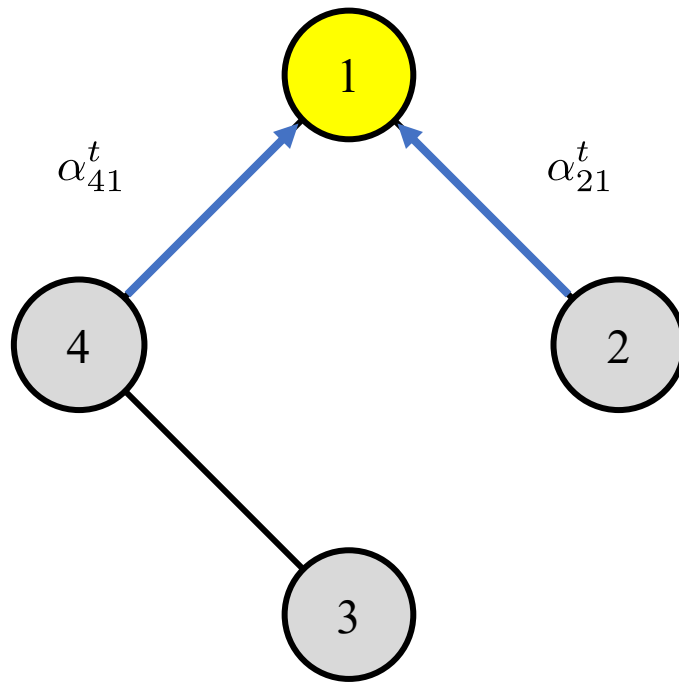
$$\mathbf{h}_i^{t+1} = \mathbf{h}_i^t + \text{MLP}_t(\bar{\mathbf{m}}_i^t)$$

Residual addition requires matching dimensions. Share the update across nodes.

Graph Attention as Message Passing

Graph attention is message passing with learned neighbor weights [11].

A shared score function sets each incoming message's contribution.



Score

$$s_{ji}^t = f_{\text{score}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

Normalize over the receiving neighborhood

$$\alpha_{ji}^t = \frac{\exp(s_{ji}^t)}{\sum_{k \in \mathcal{N}_i} \exp(s_{ki}^t)}, \quad j \in \mathcal{N}_i$$

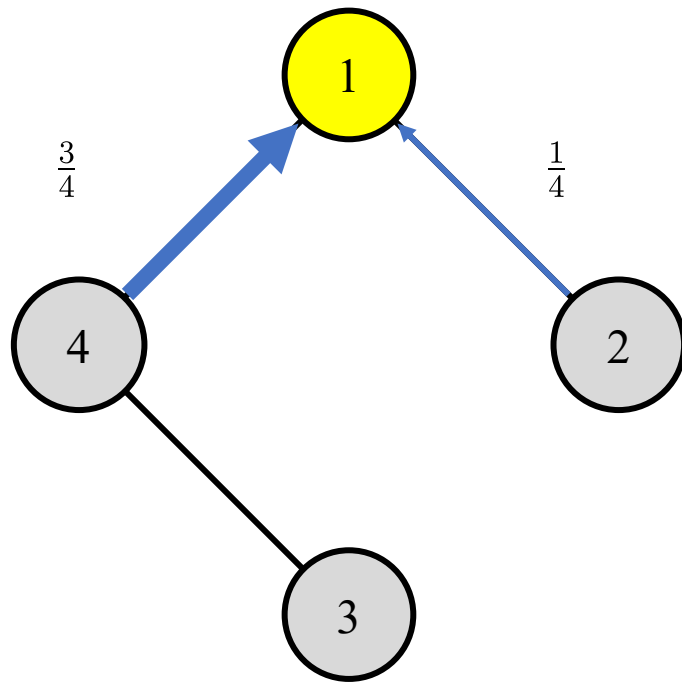
Weighted aggregation

$$\bar{\mathbf{m}}_i^t = \sum_{j \in \mathcal{N}_i} \alpha_{ji}^t \mathbf{m}_{ji}^t$$

Graph Attention as Message Passing

Graph attention is message passing with learned neighbor weights [11].

Illustrative weights on the same graph. Highlighted edges arrive at node 1.



Identity messages from the earlier node states

$$\bar{\mathbf{m}}_1^0 = \frac{1}{4}(2) + \frac{3}{4}(4) = \frac{7}{2}$$

$$B_{ij}^t = \begin{cases} \alpha_{ji}^t, & j \in \mathcal{N}_i, \\ 0, & \text{otherwise.} \end{cases}$$

		Sending node j			
		1	2	3	4
Receiver node i	1	0	1/4	0	3/4
	2	1	0	0	0
	3	0	0	0	1
	4	1/3	0	2/3	0

Rows sum to 1. Weights can differ by direction.

Outline

- Applications of Graphs
- Background
- Challenges of Deep Learning on Graphs
- GNNs
 - Overview
 - Message Passing
 - Architectures and Attention
 - **Readout**
- Implementation
- Relationship w. Transformers

Readout in GNNs

Instantiations:

1. Node Readout

$$\mathbf{y}_i = f_{\text{readout}}(\mathbf{h}_i^T)$$

2. Edge Readout

$$\mathbf{y}_{ij} = f_{\text{readout}}(\mathbf{h}_i^T, \mathbf{h}_j^T)$$

3. Graph Readout

$$\mathbf{y} = f_{\text{readout}}(\{\mathbf{h}_i^T\})$$

Readout in GNNs

Instantiations:

1. Node Readout

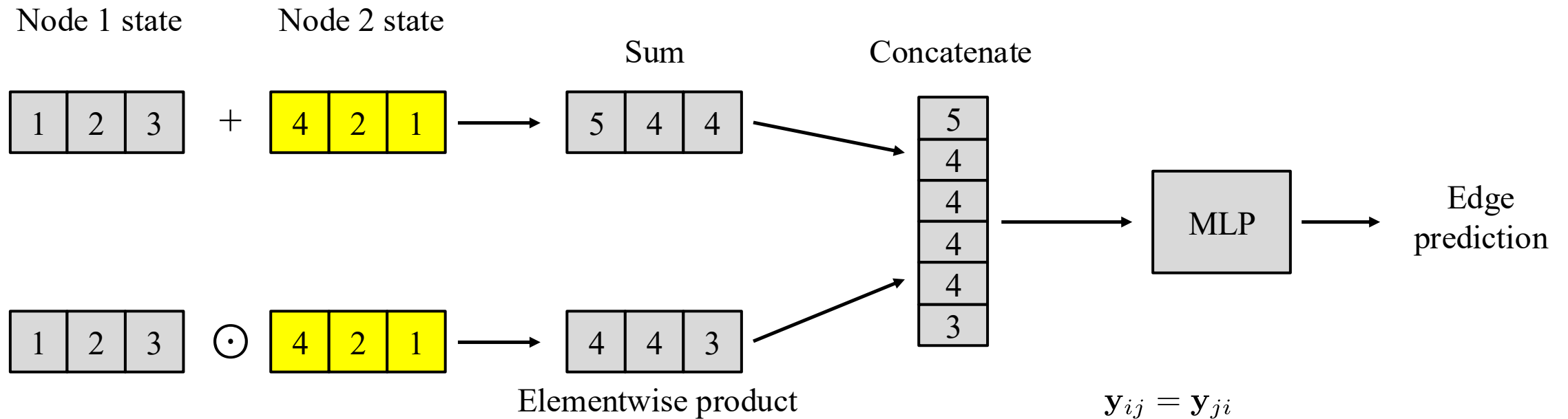
$$\mathbf{y}_i = f_{\text{readout}}(\mathbf{h}_i^T)$$

$$f_{\text{readout}}(\mathbf{h}_i^T) = \text{MLP}(\mathbf{h}_i^T)$$

Example: predict each paper's topic using the same node readout.

Readout in GNNs

For an undirected target, use a symmetric function of the two endpoint states.

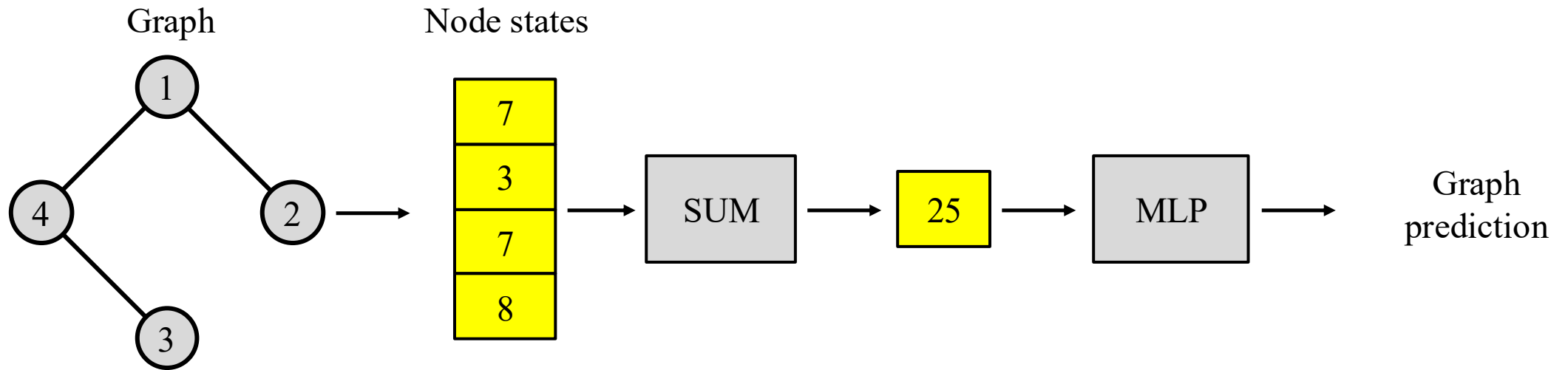


$$\mathbf{y}_{ij} = \text{MLP}([\mathbf{h}_i^T + \mathbf{h}_j^T, \mathbf{h}_i^T \odot \mathbf{h}_j^T])$$

Swapping endpoints leaves both results unchanged.

Readout in GNNs

Aggregate all node states, then predict a graph-level target.

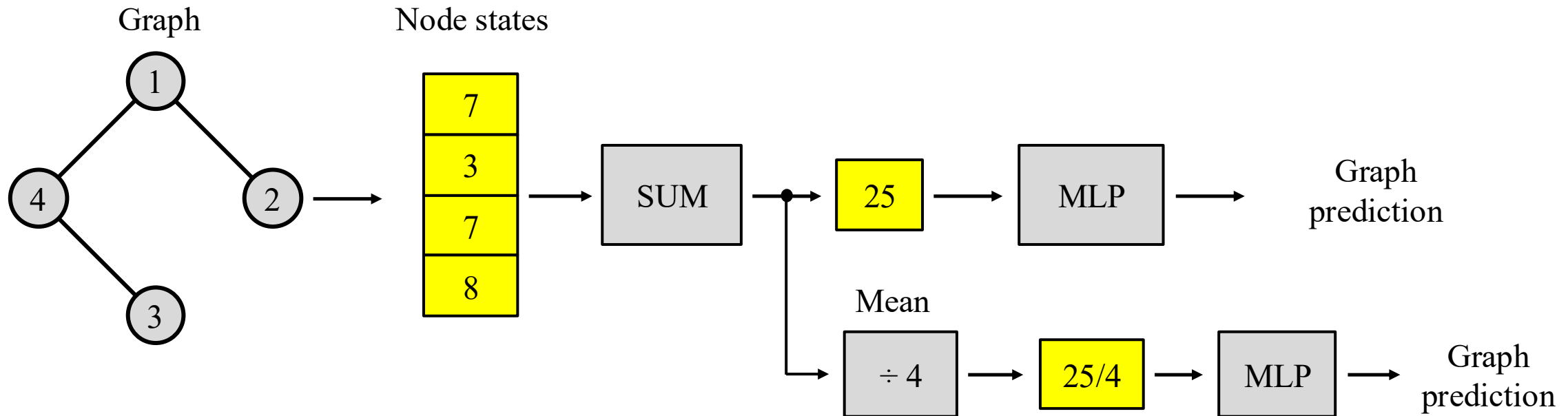


$$\mathbf{r}_{\text{sum}} = \sum_{i \in V} \mathbf{h}_i^T, \quad \mathbf{y} = \text{MLP}(\mathbf{r}_{\text{sum}})$$

Sum and mean are invariant to the ordering of the nodes.

Readout in GNNs

Aggregate all node states, then predict a graph-level target.



$$\mathbf{r}_{\text{sum}} = \sum_{i \in V} \mathbf{h}_i^T, \quad \mathbf{y} = \text{MLP}(\mathbf{r}_{\text{sum}})$$

$$\mathbf{r}_{\text{mean}} = \frac{1}{|V|} \sum_{i \in V} \mathbf{h}_i^T$$

Sum retains total scale; mean averages over nodes. The task determines which is useful.

Outline

- Applications of Graphs
- Background
- Challenges of Deep Learning on Graphs
- GNNs
 - Overview
 - Message Passing
 - Architectures and Attention
 - Readout
- **Implementation**
- Relationship w. Transformers

Implementations

1. Keep graph connectivity sparse; batch dense feature transformations.
2. Compute node and edge operations in parallel.

Implementations

1. Keep graph connectivity sparse; batch dense feature transformations [17].
2. Compute node and edge operations in parallel.

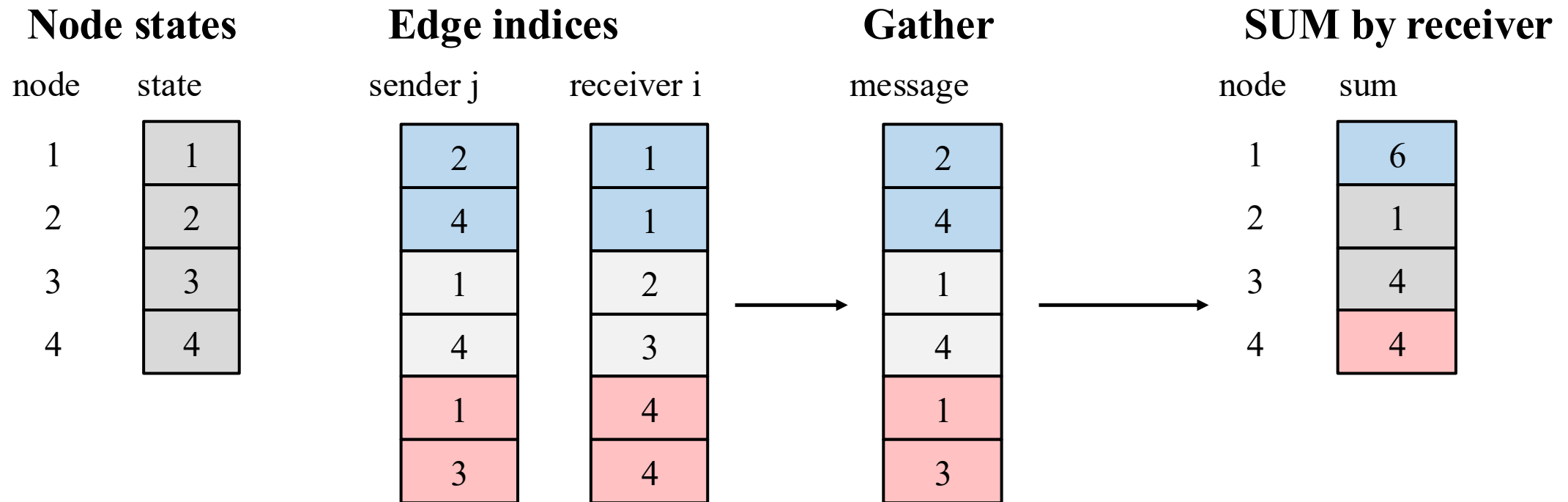
Tips:

- Store connectivity as edge indices or a sparse adjacency
- Gather node states and compute edge messages in batches
- Aggregate with scatter/segment reductions or sparse–dense multiplication
- Apply shared node updates in batches

Implementation: Gather and Reduce

The same four-node example can be evaluated in batches [17].

Store each undirected edge in both directions. Reduce messages by their receiving node.



$$\mathbf{h}_i^{t+1} = \text{ReLU}(\mathbf{h}_i^t + \bar{\mathbf{m}}_i^t)$$

→ next states: (7, 3, 7, 8)

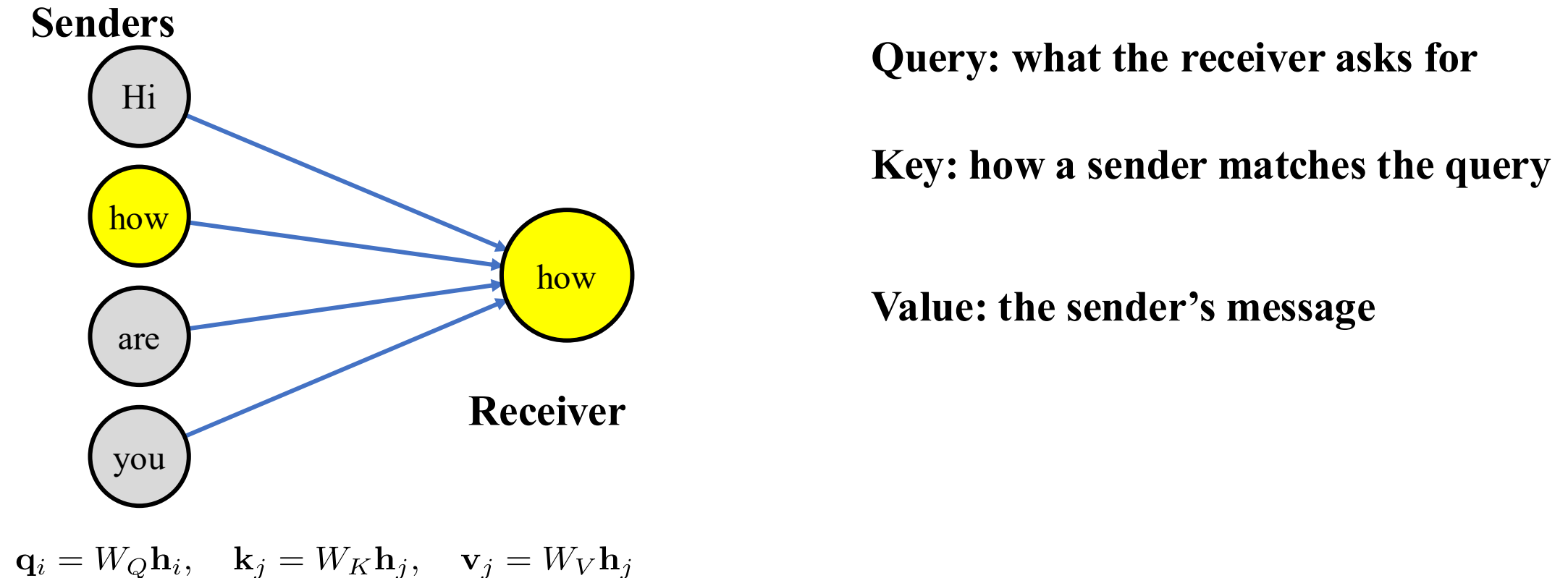
Outline

- Applications of Graphs
- Background
- Challenges of Deep Learning on Graphs
- GNNs
 - Overview
 - Message Passing
 - Architectures and Attention
 - Readout
- Implementation
- **Relationship w. Transformers**

Self-Attention as Message Passing

Revisit after the Transformer lecture

Self-attention lets each token receive messages from all tokens, including itself [10].

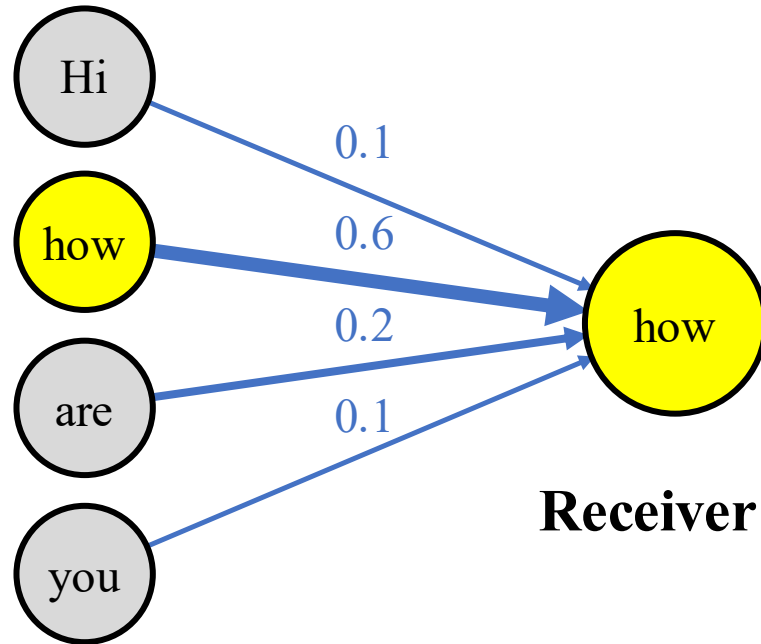


Self-Attention as Message Passing

Scores compare the receiver query with each sender key [10].

$$s_{ji} = \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_k}}, \quad \alpha_{ji} = \frac{\exp(s_{ji})}{\sum_k \exp(s_{ki})}$$

Senders



Attention weights: rows receive, columns send

	Hi	how	are	you
Hi	0.7	0.1	0.1	0.1
how	0.1	0.6	0.2	0.1
are	0.1	0.3	0.5	0.1
you	0.1	0.3	0.3	0.3

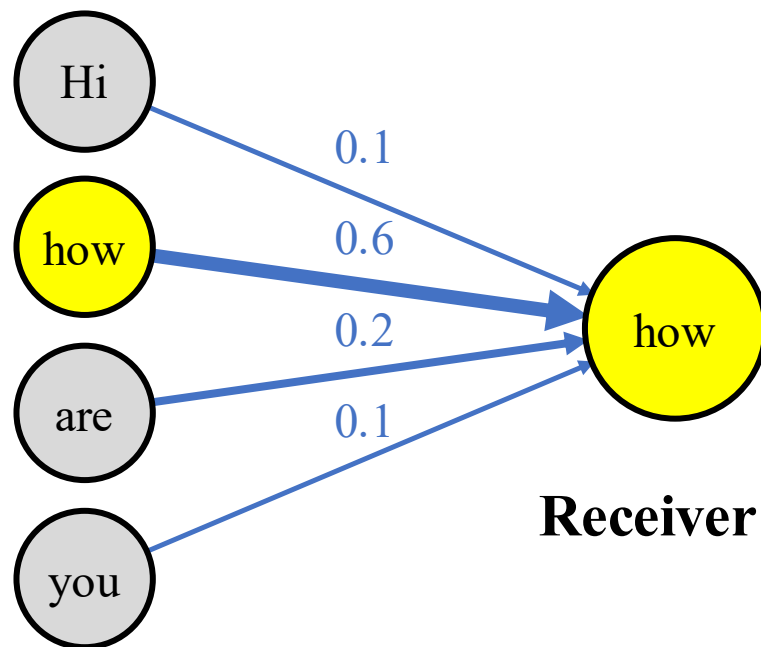
Each row is normalized over senders.

Self-Attention as Message Passing

Scores compare the receiver query with each sender key [10].

$$s_{ji} = \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_k}}, \quad \alpha_{ji} = \frac{\exp(s_{ji})}{\sum_k \exp(s_{ki})}$$

Senders



$$\mathbf{z}_{\text{how}} = 0.1\mathbf{v}_{\text{Hi}} + 0.6\mathbf{v}_{\text{how}} + 0.2\mathbf{v}_{\text{are}} + 0.1\mathbf{v}_{\text{you}}$$

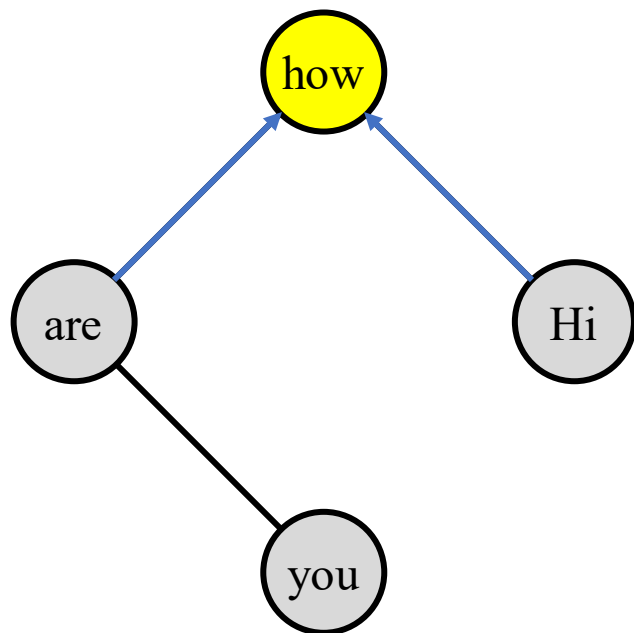
Attention weights: rows receive, columns send

	Hi	how	are	you
Hi	0.7	0.1	0.1	0.1
how	0.1	0.6	0.2	0.1
are	0.1	0.3	0.5	0.1
you	0.1	0.3	0.3	0.3

Values are messages; the weighted sum is aggregation.

Graph Structure in Attention

Restrict messages to graph neighbors with an attention mask [11].



Attention received by “how”

	Hi	how	are	you
Dense	0.1	0.6	0.2	0.1
Neighbor mask	1	0	1	0
Renormalize	1/3	0	2/3	0

Keep Hi and are; divide their weights by 0.3.

Alternatively, graph distances can bias scores without removing edges [12].

Questions?

Extended Material

Graph automorphisms

Hierarchical graph pooling

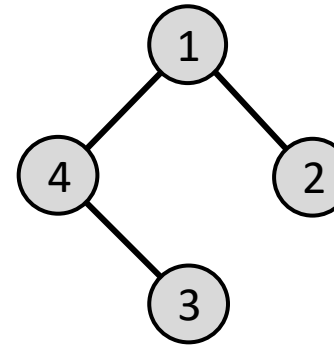
Alternative message-passing schedules

Gated graph readout

Deep Learning on Graphs

Permutation

$$\begin{array}{lll} V = [1,2,3,4] & \Rightarrow & V' = [4,3,2,1] \\ E = [(1,2), (1,4), (4,3)] & \Rightarrow & E' = [(4,3), (4,1), (1,2)] \end{array}$$



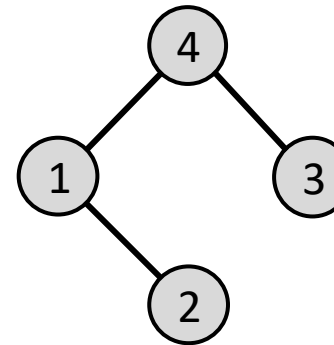
	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

Graph Automorphism:

A permutation σ of the vertex set V , such that the pair of vertices (u,v) form an edge **iff** the pair $(\sigma(u),\sigma(v))$ also form an edge.

$$PAP^{\top} = A$$

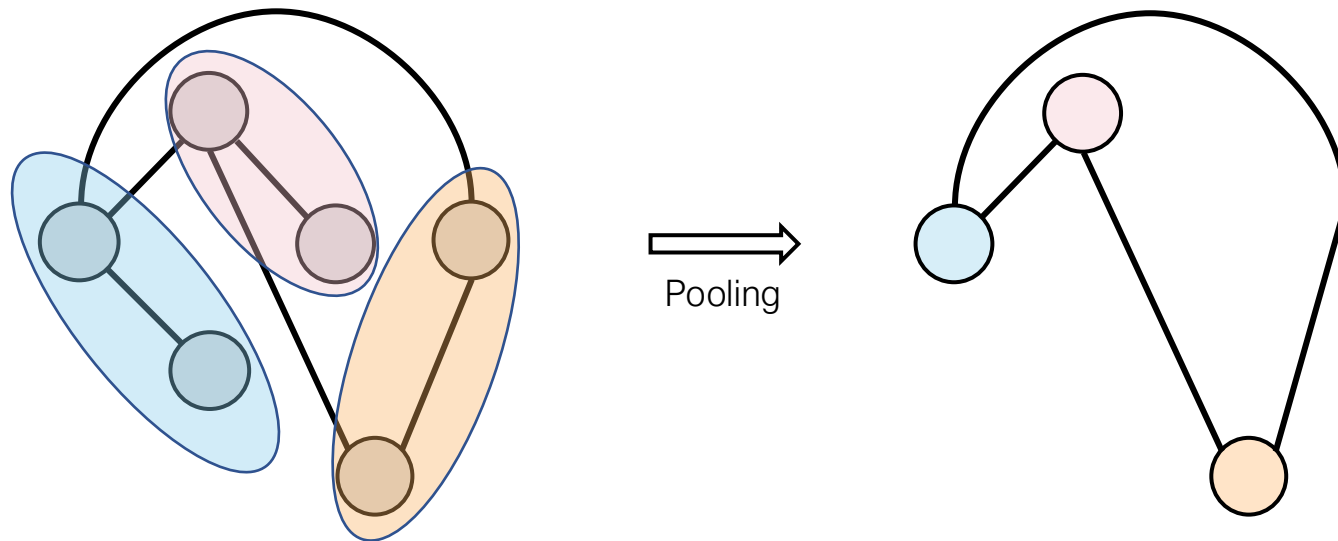


	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V' = [4,3,2,1], E' = [(4,3), (4,1), (1,2)]$$

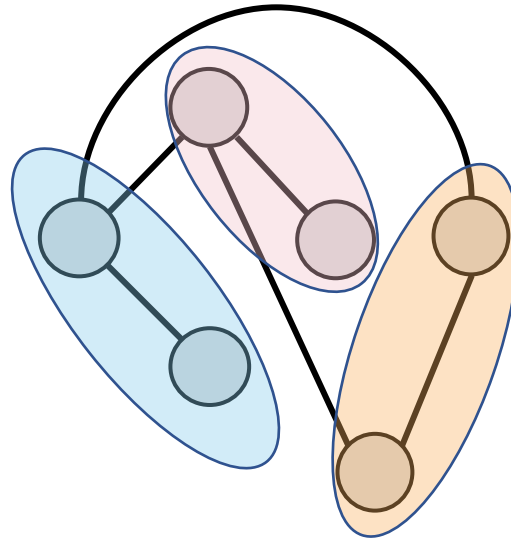
Hierarchical Graph Pooling

Merge each node group into one supernode. Keep edges between different groups.

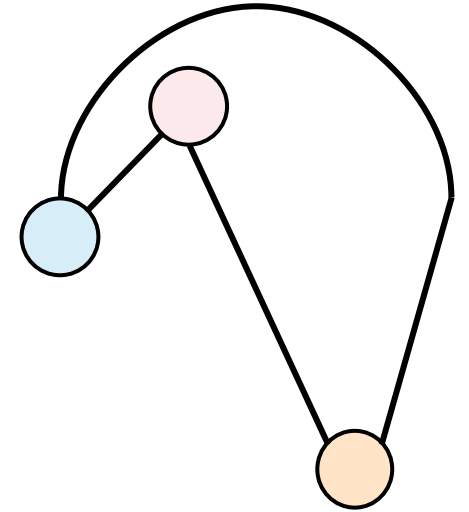


Hierarchical Graph Pooling

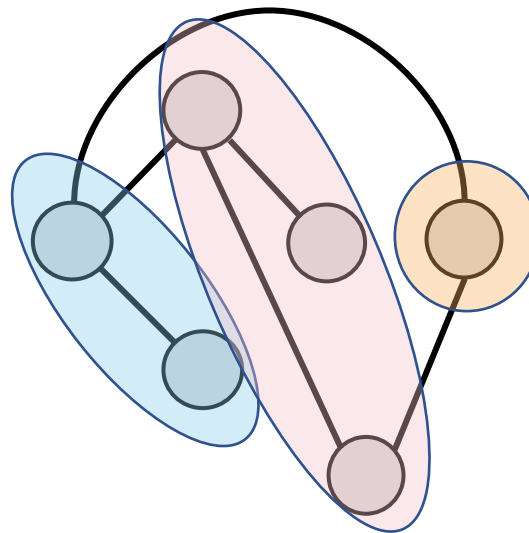
Merge each node group into one supernode. Keep edges between different groups.



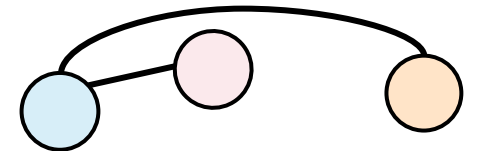
Pooling



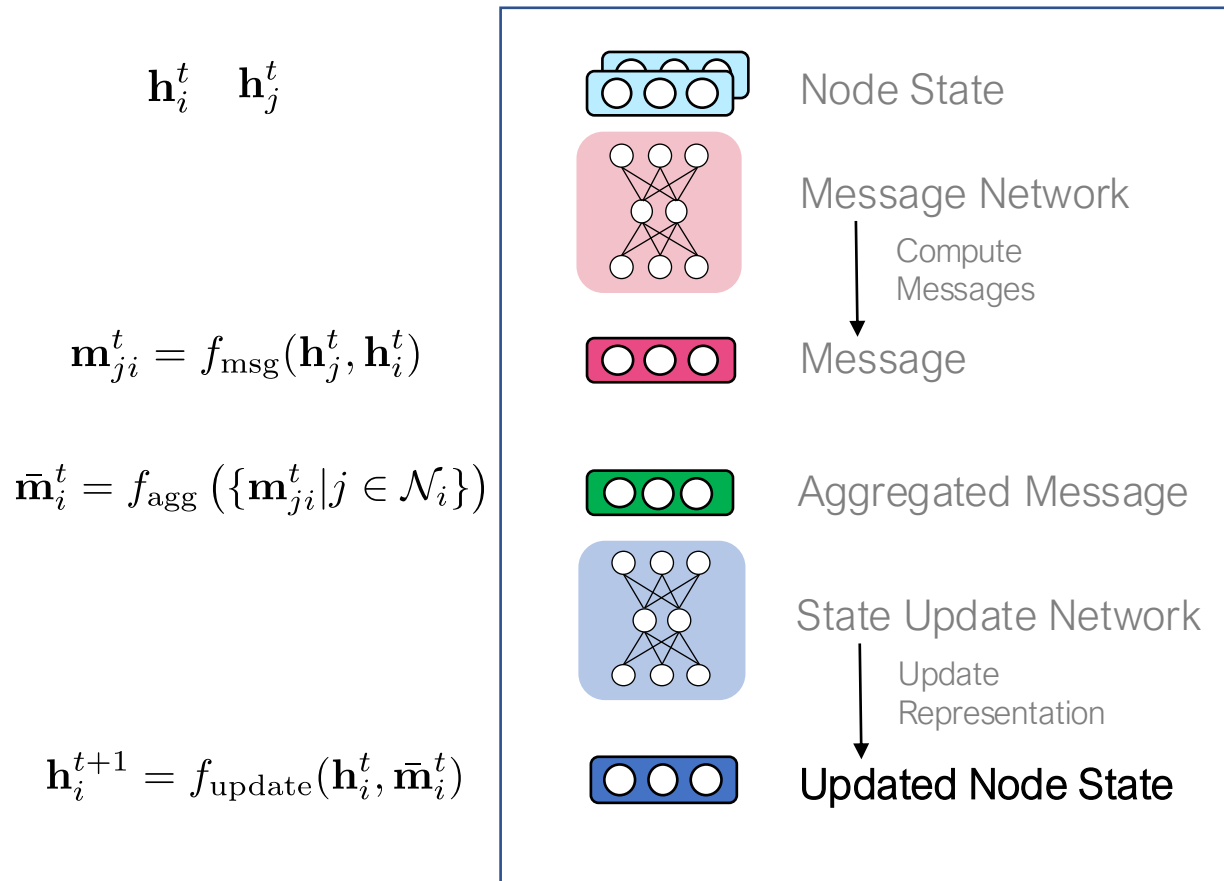
Different partitions can give different coarsened graphs.



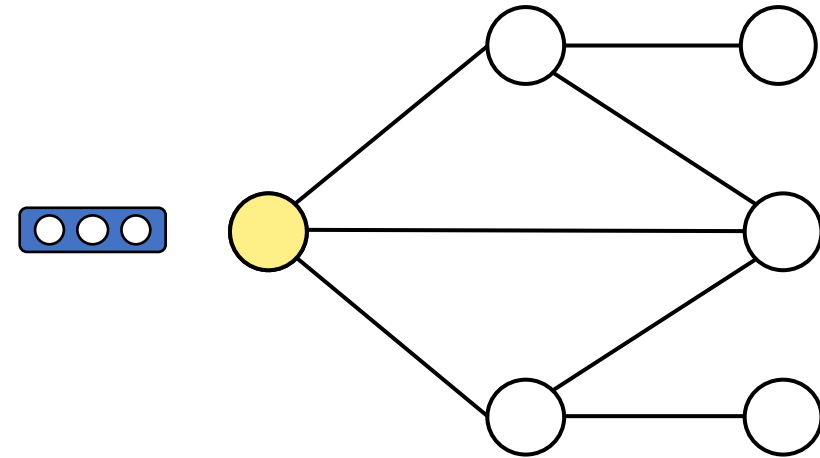
Pooling



Message Passing in GNNs



(t+1)-th message passing step/layer

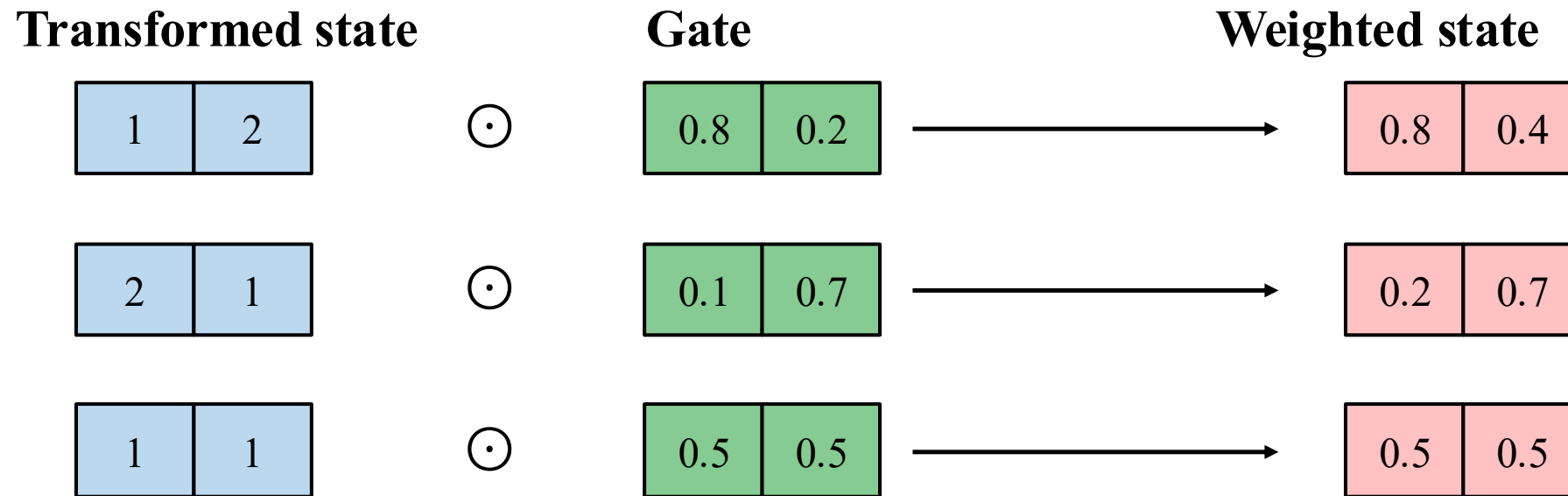


- **Parallel Schedule!**
- **Other schedules [6] are possible and could improve performance in certain tasks!**

Gated Graph Readout

Learn how strongly each node contributes to the graph representation [5, 9].

$$\mathbf{a}_i = \sigma(\text{MLP}_1(\mathbf{h}_i^T)), \quad \mathbf{u}_i = \text{MLP}_2(\mathbf{h}_i^T)$$



$$\mathbf{r} = \sum_{i \in V} \sigma(\text{MLP}_1(\mathbf{h}_i^T)) \odot \text{MLP}_2(\mathbf{h}_i^T)$$

Sum over nodes, then predict from the graph representation.

References

- [1] <https://www.tudelft.nl/en/library/research-analytics/case-12-citation-networks-2>
- [2] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M. and Monfardini, G., 2008. The graph neural network model. *IEEE transactions on neural networks*, 20(1), pp.61-80.
- [3] Goller, C. and Kuchler, A., 1996, June. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96)* (Vol. 1, pp. 347-352). IEEE.
- [4] Ackley, D.H., Hinton, G.E. and Sejnowski, T.J., 1985. A learning algorithm for Boltzmann machines. *Cognitive science*, 9(1), pp.147-169.
- [5] Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O. and Dahl, G.E., 2017, July. Neural message passing for quantum chemistry. In *International conference on machine learning* (pp. 1263-1272). PMLR.
- [6] Liao, R., Brockschmidt, M., Tarlow, D., Gaunt, A.L., Urtasun, R. and Zemel, R., 2018. Graph partition neural networks for semi-supervised classification. *arXiv preprint arXiv:1803.06272*.
- [7] Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G. and Grohe, M., 2019, July. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 4602-4609).
- [8] Hamilton, W.L., Ying, R. and Leskovec, J., 2017, December. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 1025-1035).

References

- [9] Li, Y., Tarlow, D., Brockschmidt, M. and Zemel, R., 2015. Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493.
- [10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).
- [11] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. and Bengio, Y., 2017. Graph attention networks. arXiv preprint arXiv:1710.10903.
- [12] Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y. and Liu, T.Y., 2021. Do Transformers Really Perform Bad for Graph Representation?. arXiv preprint arXiv:2106.05234.
- [13] Gori, M., Monfardini, G. and Scarselli, F., 2005. A new model for learning in graph domains. IJCNN, pp.729–734.
<https://doi.org/10.1109/IJCNN.2005.1555942>
- [14] Merchant, A., et al., 2023. Scaling deep learning for materials discovery. Nature, 624, pp.80–85.
<https://www.nature.com/articles/s41586-023-06735-9>
- [15] Ying, R., et al., 2018. Graph convolutional neural networks for web-scale recommender systems. KDD, pp.974–983.
<https://arxiv.org/abs/1806.01973>

References

- [16] Lange, O. and Perez, L., 2020. Traffic prediction with advanced Graph Neural Networks. DeepMind. <https://deepmind.google/blog/traffic-prediction-with-advanced-graph-neural-networks/>
- [17] PyTorch Geometric. Memory-efficient aggregations. https://pytorch-geometric.readthedocs.io/en/latest/advanced/sparse_tensor.html
- [18] Bruna, J., Zaremba, W., Szlam, A. and LeCun, Y., 2014. Spectral networks and locally connected networks on graphs. International Conference on Learning Representations. <https://arxiv.org/abs/1312.6203>
- [19] Defferrard, M., Bresson, X. and Vandergheynst, P., 2016. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in Neural Information Processing Systems, 29. https://papers.neurips.cc/paper_files/paper/2016/hash/04df4d434d481c5bb723be1b6df1ee65-Abstract.html
- [20] Kipf, T.N. and Welling, M., 2017. Semi-supervised classification with graph convolutional networks. International Conference on Learning Representations. <https://arxiv.org/abs/1609.02907>
- [21] Mirhoseini, A., et al., 2021. A graph placement methodology for fast chip design. Nature, 594, pp.207–212. <https://doi.org/10.1038/s41586-021-03544-w>